COMPREHENSIVE APPROACH FOR DATA WAREHOUSE SCHEMA DESIGN AND MANAGEMENT

A THESIS

submitted to Pondicherry University in partial fulfilment of the requirements for the award of the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

M.THENMOZHI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PONDICHERRY ENGINEERING COLLEGE
PUDUCHERRY – 605 014
INDIA

JANUARY 2015

PONDICHERRY UNIVERSITY
PUDUCHERRY – 605 014

CERTIFICATE

This is to certify that this thesis titled **"COMPREHENSIVE APPROACH FOR DATA WAREHOUSE SCHEMA DESIGN AND MANAGEMENT"** submitted by **Mrs. M.THENMOZHI,** Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India for the award of the degree of **Doctor of Philosophy in Computer Science and Engineering** is a record of bonafide research work carried out by her under my guidance and supervision.

This work is original and has not been submitted, in part or full to this or any other University / Institution for the award of any other degree.

Place: Puducherry
Date:

Dr. K.VIVEKANANDAN
(SUPERVISOR)
Professor
Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry – 605 014
India.

DECLARATION

I hereby declare that this thesis titled **"COMPREHENSIVE APPROACH FOR DATA WAREHOUSE SCHEMA DESIGN AND MANAGEMENT"** submitted to the Department of Computer Science and Engineering, Pondicherry Engineering College, Puducherry, India for the award of the degree of **Doctor of Philosophy in Computer Science and Engineering** is a record of bonafide research work carried out by me under the guidance and supervision of **Dr. K. Vivekanandan, Professor, Department of Computer Science and Engineering, Pondicherry Engineering College** and that this has not formed the basis for the award of any other degree by any University / Institution before.

Place: Puducherry M.THENMOZHI

Date:

ABSTRACT

Data warehousing provides an excellent approach for any organization in transforming operational data into useful and reliable information to support the decision making process. The data warehouse must be structured according to the multidimensional format in order to facilitate business analysis by Online Analytical Processing (OLAP) or data mining tools. This multidimensional schema of the data warehouse allows an organization to have a business-oriented view of the data.

Though, a number of research works have been carried out on how a multidimensional schema should be designed and managed, there is no systematic, well structured and comprehensive design process available yet. Hence, the overall objective of this research is to study the issues related to the design and management of the data warehouse schema and provide solutions that would assist in the improved design process.

The design of the multidimensional schema in the literature is carried out either from the business requirements or data source. In several cases, the requirements are not well captured or the data source is not well understood. This may lead to several rounds of reconciliation and redesign. The existing work, though tried to automate the design process, they do not fully utilize the knowledge in the requirements and the source. Hence, in this research an ontology based multidimensional (OntoMD) schema design approach has been proposed. The OntoMD approach follows a hybrid methodology to reconcile both the requirements and the source. The proposed work generates the multidimensional schema automatically covering different phases of the data warehouse design. A design tool has been developed to carry out the steps of the OntoMD approach. To illustrate the proposed approach it has been applied to a case study, and the output is evaluated using schema quality metrics.

The data warehouse schema may evolve during the design or at later stage of implementation The reasons for evolution are due to changes in business

requirements and the autonomous nature of the data sources. The existing works mainly handle the schema changes at the physical level, and hence results in high maintenance cost. And moreover, the impact of changes over the schema has not been much investigated. To address these problems an ontology supported evolution approach (OntoEvol) has been proposed in this research. The OntoEvol approach allows the automatic restructuring of the data warehouse schema, when requirements or the source evolves. This approach also analyzes the impact of a change and performs automatic adaptation of the dependent entities. OntoEvol has been applied to a case study and also it is evaluated for its effectiveness to propagate changes and its efficiency to automatically adapt the dependent entities.

To enhance the performance of the queries, the data warehouse schema needs to be partitioned during the design. Existing approaches on data warehouse schema partitioning provide algorithms for selecting optimal fragments or partitions. Other issues such as the dimension table selection, attributes selection, fragmentation of big dimension and partition management has only been partially explored. To solve these issues an optimized partitioning approach (ORP) has been proposed in this research. The ORP provides horizontal and mixed partitioning, and also it enables the partitions to be managed in case of query evolution. A case study has been used to illustrate and evaluate the ORP approach. The obtained results show that the application of ORP over the given data warehouse results in improved query performance.

Several comparisons are made for the proposed OntoMD, OntoEvol and ORP approaches with the existing methods. Here, the experimental results show a significant improvement in the performance on various parameters. Finally, the limitations of this research are identified and presented for further research.

ACKNOWLEDGEMENT

I thank all those who directly or indirectly contributed to the completion of this thesis. It is a pleasure to convey my gratitude to them all in my humble acknowledgment.

First, I thank The Almighty for blessing, protecting and guiding me throughout my research work.

I would like to express my deepest sense of gratitude to my supervisor **Dr. K. Vivekanandan**, Professor, Department of Computer Science and Engineering, Pondicherry Engineering College who offered his constant guidance, support, and untiring help throughout the course of this thesis. I also thank him for his continuous advice and encouragement during my research activities.

I sincerely thank **Dr. D. Govindarajulu**, Principal, Pondicheny Engineering College and **Dr. V. Prithiviraj**, Former Principal, Pondicherry Engineering College for granting permission and rendering support to carry out the research work.

I am grateful to my Doctoral Committee members **Dr. S. Saraswathi**, Professor and Head, Department of Information Technology, Pondicherry Engineering College and **Dr. S. Siva Sathya**, Associate Professor, Department of Computer Science, Pondicherry University for their constructive comments on this thesis.

I would like to express my thanks to **Dr. S. Himavathi**, Dean (Research), Pondicherry Engineering College, for providing a peaceful working environment during the research work in this institute.

My sincere thanks to **Dr. D. Loganathan**, Professor and Head, and former Head of the Department, Professor **Dr. N. Sreenath**, Department of Computer

Science and Engineering, Pondicherry Engineering College, for providing a peaceful working environment and for their moral support in pursuing this research.

I also thank my **colleagues** of Department of Computer Science and Engineering and Department of Information Technology, Pondicherry Engineering College for their valuable suggestions.

I specially thank **Dr. P. Salini** and **Ms. J. I. Sheeba** for being very good friends. Their critical remarks and suggestions have always been very helpful in improving my skills and for strengthening my research work.

I acknowledge all my students for their help and support with a special mention to **G. Gayathree**, **R. Anandaraj** and **S. Rajasri**.

I would also thank the **Technical, Non-Technical** staff members of Department of Computer Science and Engineering, Pondicherry Engineering College, for their support and cooperation.

My heartfelt gratitude to my parents, **Mr. S. Muruganandam** and **Mrs. M. Anbukkarasi** who supported me the most through all my life and my education.

Most importantly, I thank my beloved husband **A. Shunmugasundaram** who is the source of inspiration for pursuing this research and my kids **S.Thanuja** and **S.Srijith** for their love, which helped me to progress in my work.

Finally, I thank all my other friends and well wishers for their motivation during my research.

M.Thenmozhi

TABLE OF CONTENTS

CHAPTER NO. **TITLE** **PAGE NO.**

CHAPTER NO. **TITLE** **PAGE NO.**

LIST OF TABLES

TABLE NO. **TITLE** **PAGE NO.**

.

LIST OF FIGURES

FIGURE NO. **TITLE** **PAGE NO.**

FIGURE NO. **TITLE** **PAGE NO.**

LIST OF ABBREVIATIONS

AMDO	-	Automating Multidimensional Design from Ontologies
API	-	Application Programming Interface
CAA	-	Cost of Automated Adaptation
ChAO	-	Change Annotation Ontology
CMA	-	Cost of Manual Adaptation
DBGEN	-	Data Generator
DBMS	-	Database Management System
DDL	-	Data Definition Language
DSO	-	Data Source Ontology
DW	-	Data Warehouse
DWA	-	Data Warehouse Administrator
DWE	-	Data Warehouse Evolution
DWO	-	Data Warehouse Ontology
DWRO	-	Data Warehouse Requirement Ontology
ETL	-	Extract Transform Load
GAHC	-	Genetic and Hill Climbing
GEM	-	Generating ETL and Multidimensional designs
GUI	-	Graphical User Interface
I/O	-	Inputs and Outputs
IDE	-	Integrated Development Environment
LUMB	-	Leigh University Benchmark
MDC	-	Multidimensional Concepts
MVTDW	-	Multiversion Trajectory Data Warehouse
OBDW	-	Ontology based Data Warehouse
OLAP	-	Online Analytical Processing
OntoEvol	-	Ontology based Evolution Approach
OntoMD	-	Ontology based Multidimensional Design
ORP	-	Optimization of Referential Partitioning

ORP-H	-	Optimization of Referential Partitioning - Horizontal
ORP-M	-	Optimization of Referential Partitioning - Mixed
OWL	-	Web Ontology Language
PA	-	Partition Attributes
PT	-	Partition Tables
QAM	-	Query Attribute Matrix
QAM	-	Query Attribute Matrix
RDF	-	Resource Description Framework
SQL	-	Structure Query Language
SSB	-	Star Schema Benchmark
TPC-H	-	Transaction Processing Council Benchmark H
TPP	-	Transaction Processing Performance
UML	-	Unified Modeling Language
XML	-	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The data warehouse (DW) has been extensively used by large organizations for business analysis. It allows the top management to take critical decisions in order to improve their business. In support of the decision making process the DW integrates several heterogeneous data sources in multidimensional structures. As the DW involves a multifaceted environment, the design of the multidimensional schema and its management is a complex task, which is the focus of this research.

This chapter gives an introduction to the basic concepts underlying in this research and the preface about the research work. Section 1.2 discusses the overview of DW system. The detailed information about the life cycle of a DW is given in section 1.3. Section 1.4 provides the details about the multidimensional model which represents the DW schema. Section 1.5 discusses the DW schema design and management aspects. The use of ontology for a DW is explained in section 1.6. Section 1.7 provides the motivation behind this research. The problem statement is described in section 1.8. Section 1.9 provides the objective of this research work. Section 1.10 outlines the research contribution. Finally, section 1.11 provides the organization of this thesis.

1.2 AN OVERVIEW OF DATA WAREHOUSE SYSTEM

Information is a very powerful asset that can provide significant benefits to any organization and a competitive advantage in the business world. The organizations have vast amounts of data but have found it increasingly difficult to access it and make use of it. The reason is that, the data is in many different formats, exists on many different platforms, and resides in many different file and database

structures (Poe *et al.,* 1997). Hence, organizations have to write and maintain hundreds of programs that are used to extract, arrange, and integrate data for use by many different applications for analysis and reporting. This would typically require modification of the extract programs or development of new ones. The data warehousing provides an excellent approach for transforming the vast amounts of data that exist in these organizations into useful and reliable information (March and Hevner, 2007). For business executives the data warehousing provides architectures and tools to systematically organize, as well as understand, and use their data to make strategic decisions (Cho and Ngai, 2003). It supports information processing by providing a solid platform of consolidated and historical data for analysis. Such systems are valuable tools in today's competitive and fast-evolving world.

The DWs have been defined in many ways, making it difficult to formulate a rigorous definition. The classical definition given by W.H. Inmon is "A DW is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of the management decision-making process" (Inmon, 2005) . The four key words, subject-oriented, integrated, time-variant, and nonvolatile, explained below, distinguish DWs from other data repository systems, such as relational database systems, transaction processing systems, and file systems.

- **Subject oriented:** A DW is organized around major subjects, such as orders, customers, and sales. It mainly focuses on the modeling and analysis of data for decision makers. Moreover, it provides a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process.
- **Integrated:** The DW is constructed by integrating multiple, heterogeneous data sources such as relational databases, on-line transaction records and flat files. During integration the data are cleaned. This ensures consistency in naming conventions, attribute measures and encoding structures among different data sources.

- **Time-variant:** The time horizon for the DW is significantly longer than that of operational database systems. That is, the operational database contains the current value of the data, whereas the DW data provide information from a historical perspective.
- **Nonvolatile:** The operational update of data does not occur in the DW environment and it does not require recovery, and concurrency control mechanisms. The two operations supported in DWs are: loading of data and access of data.

1.2.1 Applications of Data Warehouse

Many organizations use DW information to support business decision making activities (Chaudhuri and Dayal, 1997), which includes the following:

1) The analysis of customer buying patterns;
2) Repositioning products and managing product portfolios by comparing the performance of sales by time and geographic regions in order to fine-tune production strategy;
3) Analyzing business operations and looking for sources of profit;
4) Handling the customer relationships by making environmental corrections and managing the cost of corporate assets.

A DW is also very useful from the point of view of heterogeneous database integration. Various organizations typically collect diverse kinds of data and maintain large databases from multiple, heterogeneous, autonomous and distributed information sources. It is challenging to integrate such data, and provide easy and efficient access to it (Chaudhuri and Dayal, 1997). Much effort has been spent in the database industry and research community towards achieving this goal.

1.2.2 Comparison of Operational Database System and Data Warehouse

DWs have the distinguishing characteristic that they differ from the operational database systems. The major task of operational database systems is to perform online transaction and query processing (Chaudhuri and Dayal, 1997). Such systems are called online transaction processing (OLTP) systems. The day-to-day operations of an organization, such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting are carried out by these systems. In contrast a DW system serves users or knowledge workers in the role of data analysis and decision making. For example, in financial services the DW can be used for risk analysis, credit card analysis, and fraud detection. These systems can organize and present data in various formats in order to accommodate the diverse needs of the different users. Such systems are known as decision support systems (Berson and Smith, 1997).

An operational database system contains current raw data, such as transactions, which need to be consolidated before analysis. Whereas, the DW contains historical data which are consolidated (such as aggregation and summarization) from heterogeneous sources to facilitate business analysis (Conn, 2005). Moreover, the tables are in normalized form in operational database but the DW contains de-normalized tables to provide fewer joins in order to improve the query performance. Short and fast inserts and updates are initiated by end users in case of operational database. Periodic long-running batch jobs refresh the data within the DW.

An operational database is designed and tuned from known tasks and workloads, for example, indexing and hashing using primary keys, searching for particular records, and optimizing queries (Bog *et al.*, 2011). On the other hand, DW queries are often complex and involve the computation of large groups of data at a summarized level. Thus, the separation of operational database systems from DW is based on the difference in structure, content, and the use of data in these two

systems. Since the two systems provide quite different functionalities and require different kinds of data, it is necessary to maintain them as separate systems (Chaudhuri and Dayal, 1997; Kalnis and Papadias, 2001). The Table 1.1 summarizes the difference between an operational database system and a DW.

Table 1.1 Differences between Operational System and Data Warehouse

Operational Database System	Data Warehouse
Transaction Oriented	Subject Oriented
Small (MB upto several GB)	Large (GB upto several TB)
Current Data	Historical Data
Normalized Table Structure	De-Normalized Table Structure
Continuous Updates	Batch Updates
Simple to Complex queries	Usually Complex queries

1.2.3 Data Warehouse Architecture

The DW architecture encapsulates all the facets of an enterprise environment. The architectures vary depending upon the specifics of an organization's situation. The data for the DW come from operational systems of the organization as well as from other external sources. It is populated with the data that are extracted from operational systems and stored in an area called data staging area, which are then cleaned, transformed and integrated. A presentation server is the target machine on which the data is loaded from the data staging area. Here the data is organized and stored for direct querying by end users, report writers and other applications. Each component of the architecture (Kimball and Ross, 2002) is represented in the Figure 1.1 and the tasks performed by them are explained below:

- **Operational Source Systems** These are the systems of record that capture the transactions of the business. The main priorities of the operational systems are processing performance and availability. The queries imposed over such systems are simple, that are part of the normal transaction flow and severely restricted in their demands on the operational system (Kimball and Ross, 2002).

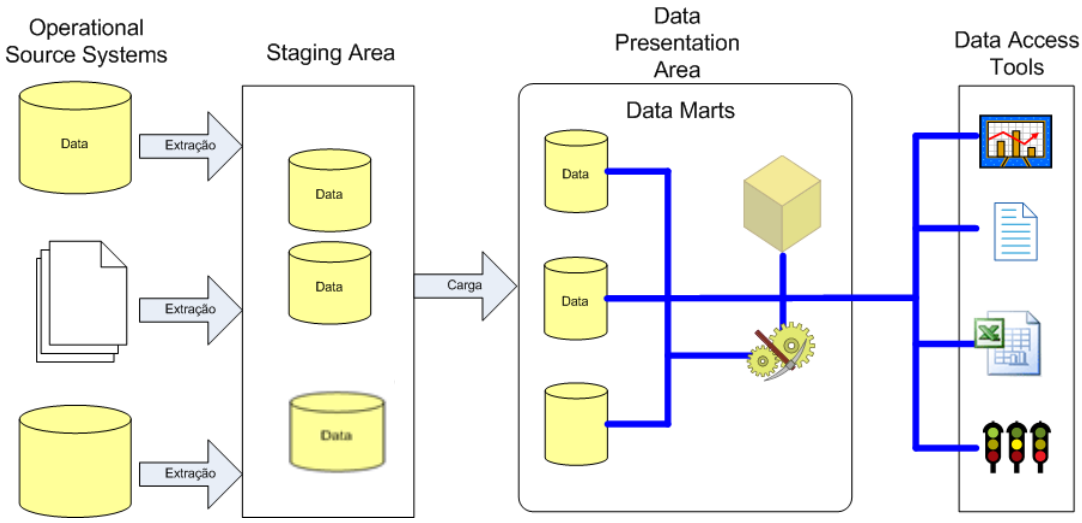

Figure 1.1 Data Warehouse Architecture

- **Data Staging Area** A data staging area is where the raw operational data are extracted, cleaned, transformed (ETL) and combined so that it can be reported on and queried by users (Kimball and Ross, 2002). The data staging area lies between the operational source systems and the DW and is typically not accessible to users. Extraction is the first step in the process of getting data into the DW environment which involves reading and understanding the source data and copying the data needed in the staging area for further manipulation. Once the data is extracted in the staging area, there are numerous potential transformations, such as cleansing the data, combining data from multiple sources, de-duplicating data, and assigning warehouse keys. The final step of the Extract, Trasform and Load (ETL) process is the loading of data in the DW environment.

- **Data Presentation** The data presentation area is where the data is organized, stored, and made available for direct querying by users, report writers, and other analytical applications. The presentation area is typically referred to as a series of integrated data marts. Data marts are smaller DWs, focusing on a small subset of the enterprise data (Kimball and Ross, 2002). Typically each data mart is used

by a particular unit of the organization for various strategic analyses relevant to its goals. Data is extracted from the corporate warehouse into the data mart periodically and used for analysis.

- **Data Access Tools** The final major component of the DW environment is the data access tool(s). All data access tools query the data in the DW's presentation area. Some of the user access tools are reporting and query tools, OLAP tools and data mining tools (Kimball and Ross, 2002).

1.3 DATA WAREHOUSE LIFE CYCLE

The various activities of a DW such as design, development and implementation are provided through the DW life cycle. This life cycle is called as the Kimball life cycle or business dimension life cycle (Kimball 1998). Here, the different tasks to be sequenced for a DW project are identified and activities that need to happen concurrently are highlighted. It is essential that the different activities in the life cycle need to be customized to address the unique needs of the organization. Figure 1.2 shows the structure of the business dimension life cycle. The different phases of the life cycle are explained below:

1. **Project Planning:** This phase includes the definition of system goals and properties, assessment of the impact of organizational practices, an estimate of costs and benefits, allocation of the required resources and a preliminary plan for the project (Kimball, 1998).
2. **Business Requirement Definition:** This phase plays a vital role in making designers fully understand the user needs in order to maximize the benefits and profitability of the system under consideration. The designers at this stage are involved in identifying the key-factors of the decision making process and convert them into design specifications (Kimball, 1998). Following the requirements three different architectures are carried out in parallel such as data, technology and application.

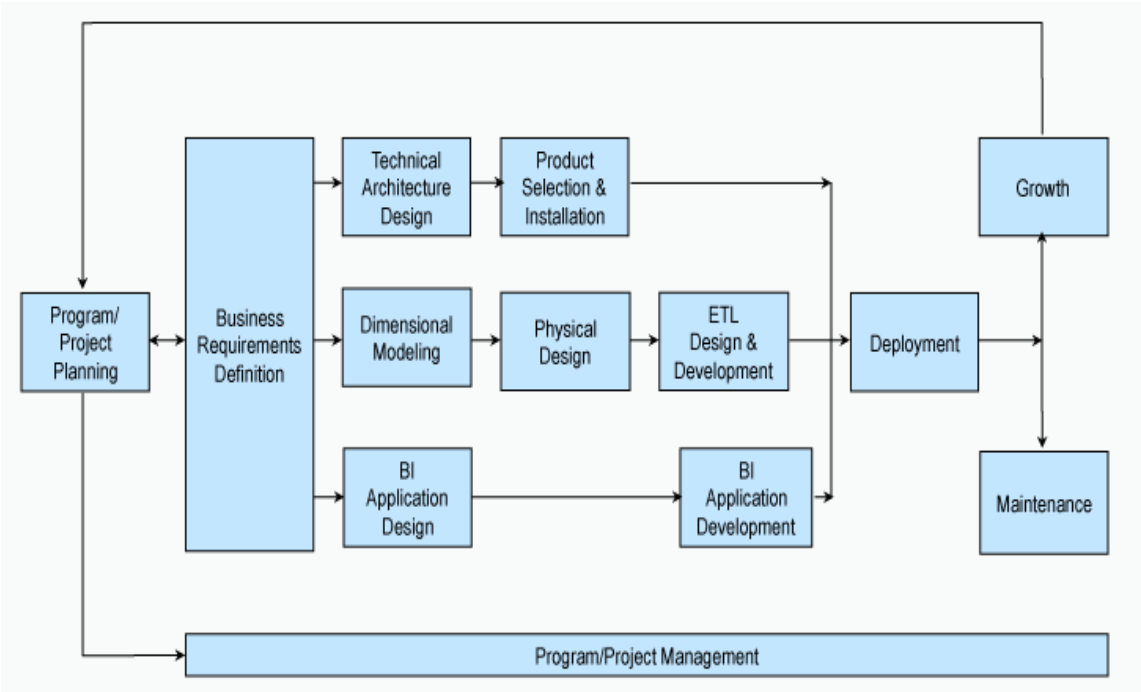

Figure 1.2 Business Dimension Lifecycle

3. **Data Architecture:** First phase of this architecture includes *dimensional modeling*, where the user requirements and analysis of the data source leads to the structure of the DW. The result of this phase is a logical model containing relationship with the source schema. The next phase of the data track is the *physical design*. Here the logical model is transformed into the physical model by considering the optimization and implementation factors related to the selected database such as indexing and partitioning (Kimball, 1998). The final phase *designing and developing data staging* involves extraction, transformation and loading of data from the source to the DW.

4. **Technology Architecture:** This architecture includes an *architectural design* phase, which is based on the current technical specification for business information systems and performance requirements set by users (Kimball, 1998). The next phase includes *product selection and installation*

of hardware platform, ETL tools, database management system, data access query tools, and reporting tools (Kimball, 1998).

5. **Application Architecture:** In this architecture, the *user application specification* phase includes a collection of specification for the application that provides end user with data access (Kimball, 1998). Along with this, the assessment of reports, interactive data navigation and automatic knowledge extraction are carried out. The next phase, *user application development* involves setup and configuration of analysis tools selected during *the product selection phase* (Kimball, 1998).
6. **Deployment:** Once the design and implementation tasks are completed this phase is used to deploy the reports, query tools, and applications to the user community (Kimball, 1998). The deployment is deferred until all the training, documentation, and validated data are available for production release. Deployment ensures the results of technology, data, and application architectures are tested and fit together properly.
7. **Maintenance and Evolution Phase:** It begins once the system is deployed into production and it ensures ongoing support with business users (Kimball, 1998). Technical operational tasks such as performance tuning, index maintenance, usage monitoring, and system backup are done periodically by technical experts. DW maintenance mainly concerns performance optimization that must be periodically carried out. On the other hand, DW evolution concerns keeping the DW schema up-to-date with respect to the business domain and the business requirement changes.

The next life cycle iteration usually begins during the deployment of the previous iteration. The whole process starts again when the business analysts and designers gathers detailed requirements for the next highest priority business process and creates the associated dimensional model. The incremental approach of the life cycle is a fundamental element that delivers business value in a short period, while building a long-term enterprise information resource (Kimball, 1998).

1.4 MULTIDIMENSIONAL MODEL

DWs have the distinguishing characteristic that they are mainly intended for decision support applications. Hence the arrangement of data within the warehouse is different from those adopted for operational information systems (Golfarelli *et al.*, 1998). For online transaction processing by an operational system, a data model such as Entity Relationship (ER) model might be appropriate. To facilitate online data analysis a DW requires a concise, subject-oriented schema. The most popular data model for a DW is a multidimensional model (Golfarelli *et al.*, 1998). It provides both a mechanism to store data and a way for business analysis. Basic components of the multidimensional model are facts, measures, dimensions and hierarchies which are explained below:

A *fact* is a focus of interest for the decision-making process of an organization. It typically corresponds to events occurring dynamically in the enterprise world such as sales or orders (Golfarelli *et al.* 1998). *Measures* are continuously valued attributes that describe the fact numerically (Golfarelli *et al.* 1998). For the business analysis their values are used for mathematic calculations that include summation, average, minimum and maximum. *Dimensions* are mutually independent parameters that describe the business process fact (Golfarelli *et al.* , 1998). Every dimension has a discrete domain of possible values.

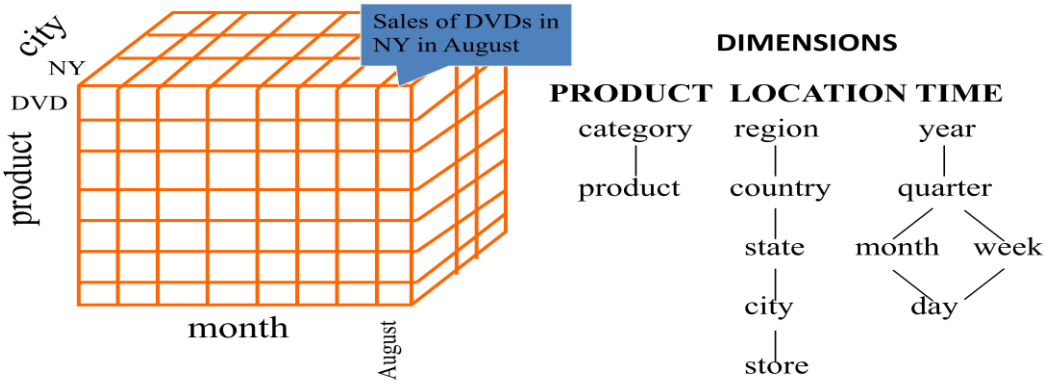

Figure 1.3 Multidimensional Model

The business process can be viewed at different levels of abstraction. Two or more levels at different level of abstraction form a hierarchy. For example, in Figure 1.3 the product name is related to its category attribute through such a hierarchical relationship. The hierarchies may also include descriptive attributes that contain additional information about a level of the hierarchy. The multidimensional model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema (Kimball and Ross, 1996; Moody and Kortnik, 2000).

- **Star schema:** The most common modeling paradigm is the star schema, in which the DW contains a large central table called *fact* containing the bulk of the data, with no redundancy, and a set of smaller dimension tables (Kimball and Ross, 1996). A star schema for sales DW is shown in Figure 1.4. The schema contains a central *fact* table for sales along with four dimensions, namely, product, time, branch, and location. The fact table contains keys for each of the four dimensions and with two measures, dollars sold and units sold. Each dimension in the star schema is represented by only one table which contains a set of attributes. For example, the item dimension table contains the attribute set product_key, product_name, product_brand, product_type and supplier_type. Moreover, the attributes within a dimension table may form a hierarchy.

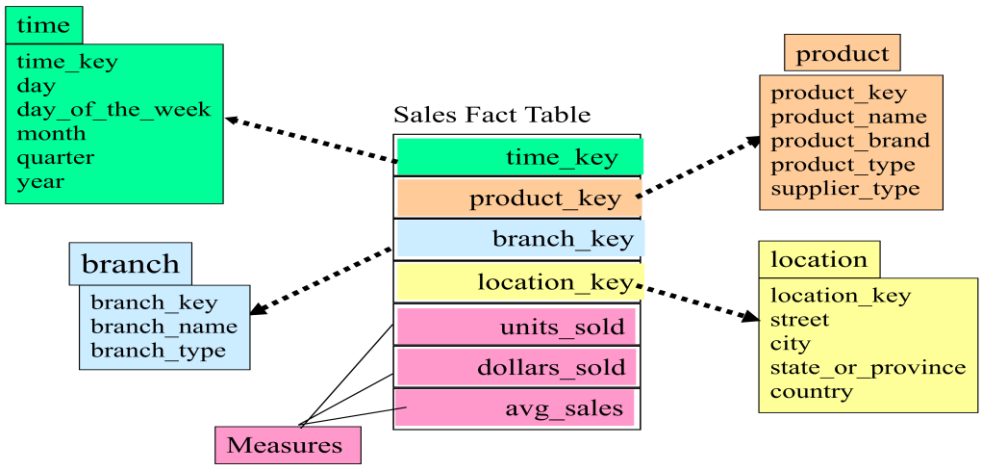

Figure 1.4 Star Schema

- **Snowflake schema:** The snowflake schema is a variant of the star schema model. The major difference between the snowflake and star schema model is that the dimension tables of the snowflake model may be kept in normalized form to reduce redundancies. A snowflake schema for sales DW is given in Figure 1.5.

The other type of schema used in DW context is the fact constellation schema. This schema allows dimension tables to be shared between multiple fact tables. It is used by few organizations since it can model multiple, interrelated subjects that span the entire organization. On the other hand a star or snowflake schema is commonly used for a data mart, which is a department subset of the DW that focuses on selected subjects.

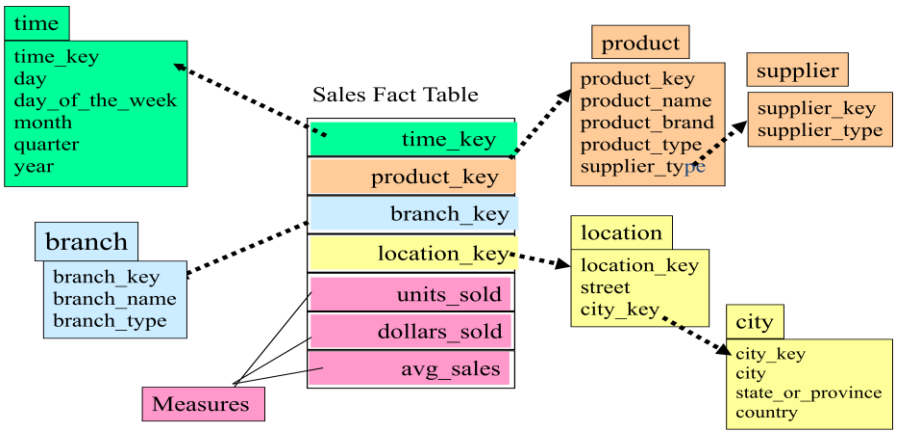

Figure 1.5 Snowflake Schema

1.5 DATA WAREHOUSE SCHEMA DESIGN AND MANAGEMENT

A DW creation process consists of five steps: pre-development activities, architecture selection, schema creation, warehouse population, and DW maintenance (Srivastava and Chen, 1999). The focus of this research is the schema creation and its management. As DW schema involves a complex structure, its design and management is different from that of the operational database system (Golfarelli and Rizzi, 1998; H¨usemann *et al.*, 2000; Luján-Mora and Trujillo, 2003).

1.5.1 Data Warehouse Schema Design

The design of the DW schema involves generating the multidimensional model involving facts, measures, dimensions and levels. To generate the multidimensional structure the designer of the DW generally follows conceptual, logical and physical phases (Golfarelli and Rizzi, 1999). And few adopt a requirement analysis as the starting phase of the design (Gardner, 1998). The conceptual phase transforms the requirements into a conceptual schema representing the multidimensional elements, the logical phase transforms the conceptual schema to a logical representation involving the table and attributes of the multidimensional structure, and the physical phase constructs the physical schema of the DW from the logical representation with implementation constraints.

The existing research work on DW schema design are mainly driven by any of the following methodologies (Winter and Strauch, 2004) to generate the DW multidimensional schema:

- **Supply-driven:** These approaches are based on analyzing the operational data sources in order to derive the multidimensional schema, while requirements are considered later when the data is about to be analyzed.
- **Demand-driven:** These approaches focus on the information needs of decision makers, and data sources are only taken into account when the data is loaded into the DW.
- **Hybrid:** These approaches advocate the consideration of both data sources and information requirements in the early stages of development.

1.5.2 Data Warehouse Schema Evolution

DW design is a continuous process and need to adapt to changes in its environment (Bellahsene, 2002). The data sources which are incorporated in the DW are autonomous in operation and they can change or evolve in terms of their

instances and schemas (Benitez-Guerrero *et al.,* 2004). Moreover, the requirements stated by the various stakeholders and developers frequently change owing to numerous reasons (Rechy-Ramirez and Benitez-Guerrero, 2006). Hence, DW needs to be managed whenever there is any change or update in the requirements or source in order to fulfill the constraints and criteria allocated by the various people who need the assistance of information preserved in the DW.

1.5.3 Data Warehouse Schema Partitioning

The DW integrates massive amounts of data from multiple sources and need to process complex analytical queries for different access forms such as OLAP, data mining and reporting tools. Hence, ensuring short query response is enormously difficult and can only be achieved by certain optimization techniques. The performance optimization techniques available in the literature are classified as i) techniques applied during the DW design ii) techniques applied after the DW is implemented (Bellatreche and Woameno 2009). Partitioning of the tables and parallel processing are two examples of the first category. Materialized views, indexes and data compression are applied during the exploitation of the DW and belongs to the second category. The use of optimization techniques belonging to the first category is more sensitive compared to those belonging to the second one as the decision of using them is usually taken at the beginning stage of the DW development (Golfarelli and Rizzi, 1999). The reason is, for instance, if the partitioning applied for a DW is not well adapted then it would be costly and time consuming to reconstitute the initial warehouse from the partitions. Whereas, the indexes or materialized views selected for a DW identified as insufficient can be dropped or replaced by other optimization techniques (Golfarelli and Rizzi, 1999). Based on sensitivity and the carefulness of techniques belonging to the first category it is essential to focus on partitioning of DW schema during the design for performance optimization.

1.6 ONTOLOGY FOR DATA WAREHOUSE DESIGN

The semantic web is increasingly seen as a powerful infrastructure to build reusable and sharable knowledge on the web (Berners-Lee *et al.,* 2001). It provides XML, RDF and OWL to describe web contents that enable automated information access on the machine processable semantics of the information and service (Gomez-Perez, 2004). Ontologies are the core of the semantic web for the reuse of formalized knowledge. Ontology is the term referring to the shared understanding of some domains of interest, which is often conceived as a set of classes (concepts), relations, functions, axioms and instances (Gruber, 1993). Ontology is most commonly defined as "a formal, explicit specification of a shared conceptualization" (Gruber, 1993). The ontology is used to solve the problem of syntactic and semantic heterogeneities that exist between different data sources (Cruz and Xiao, 2005). It is also used to analyze the knowledge related to a specific field, model the relevant concepts in a domain and facilitate the distinction of the different domain concept. Ontology can bring benefits to data warehousing developments at different phases, as it can enhance the semantics of data sources, integrate heterogeneous schemas, automate ETL process and facilitate OLAP in data analysis (Pardillo and Mazón, 2011). In recent years, researchers have proposed various approaches to bring ontology and data warehousing to solve several DW design issues.

The decision in using an ontology-based approach for DW, instead of using another technology, for example a UML-based approach, lies in the fact that ontologies may empower the automatization since they provide mechanisms to formally specify the semantics of a domain using language such as Web Ontology Language (OWL) on which models may be supported (Pardillo and Mazón, 2011). The OWL is an international standard for encoding and exchanging ontologies (Smith *et al.,* 2004). The reason for choosing the OWL is that, it provides the system with the means of not only representing information, but also for automatic processing of that information (McGuinness and Van Harmelen, 2004). Another reason is, it provides good support for reasoning. Reasoner or inference engine, is a

piece of software that can be used for automatic inference of the additional knowledge concerning the rules specified by the ontology (Wang *et al.,* 2004). Thus, the ontology may be considered as an appropriate solution to confront with the main challenge of the DW design.

1.7 MOTIVATION

The success of a DW system is dependent on the problem of designing and modeling the DW structure. Existing approaches either concentrate on user requirements or the data source for the DW schema design process. But it is often encountered that, the information contained inside the data source systems may be hidden among the multitude of data and its value is often understood only by a few top experts. Moreover, an ambiguous definition of the user requirements occurs when the users are unable to define their requirements precisely and clearly. Various meanings of data (i.e. Attributes, Tables) make it difficult for integrating the user requirements to the data sources. Thus, reconciliation of the appropriate semantics of the user terms and data sources is important in generating the DW schema accordingly.

The DW schema once designed is never meant to be static. It evolves due to changes in the data source or the requirements. As the DW is a complex environment which consists of many layers it becomes costly and labor intensive to propagate changes to the DW schema correctly and analyze its change impacts. The amount of research into the impact of DW schema changes is much less investigated. The impact of requirements changes upon DW models must often be estimated manually by application experts and moreover, no automated restructuring methods are available to the designer yet.

Another important issue related to the design and management is that, the DW need to be tuned for performance using partitioning techniques before it is populated from the data sources. In the DW context, a fact table can be partitioned based on the fragmentation schemas of dimension tables using referential

partitioning. This type of fragmentation may dramatically increase the number of fragments of the fact table and make their maintenance very costly. Moreover, the choice of dimension tables and its attributes has greater impact on the query performance. The existence of big dimension increases the complexity of partitioning. And further, the changes in the queries require alteration of existing partitions.

Thus, the motivation behind this research is that both the user requirements and data source need to be considered at the early stage of the DW schema design. And a thorough understanding of the requirements and the data source is essential, to provide a successful design. Moreover, when there are changes in the business domain it is necessary to handle the evolution of the DW schema through effective design strategies. Finally, choosing an optimal partitioned schema is crucial during the design stage, which enhances the performance of the DW after its implementation.

1.8 PROBLEM STATEMENT

In literature, some research efforts have been proposed for the automation of DW schema design using ontology. But these approaches do not fully utilize the benefits of hybrid methodology, where the user requirements and data source needs a thorough analysis. Moreover, these approaches do not cover the different phases of DW design such as conceptual, logical, and physical. Hence, it is of great importance to provide a formal, explicit, and well-defined way to represent all the parameters and properties of the user requirements and data source to guide the design task. Moreover, a full automation is essential covering all the phases of the design which helps greatly in reducing the complexity involved and reduces the dependency on an expert's ability to perform the design task.

Owing to the changes in business needs the DW schema needs to evolve. The existing approaches such as schema evolution or schema versioning handle changes either in user requirements or the data source. They mainly concentrated on

DW schema restructuring at the physical level. This may induce high maintenance costs, as any change in the DW schema structure has an impact on the dependent modules. Hence, an effective approach is required which handles both requirements and source changes as well analyze the impact of a change on the DW schema structure and its dependent modules, before it is propagated at the physical level.

The schema partitioning technique applied during the DW design optimizes the performance of the DW. As partitioning results in a large set of fragments, existing works used evolutionary algorithms to select optimal fragments. The issues related to the dimension table selection, attribute selection, optimal fragment selection big dimension and query evolution have only been partially explored. Thus, an optimal strategy is essential to improve the existing partitioning techniques.

1.9 RESEARCH OBJECTIVES

The main aim of this thesis is to provide a comprehensive approach that handles the DW schema design and management in an effective and automated way. To achieve this following are the objectives identified in the research work:

1. To provide a hybrid approach to automate the multidimensional schema design using ontology which helps to reduce the burden of the designer to perform a complex reconciliation of requirements and source, and redesign involved during the different phases of the design process.
2. To handle DW schema evolution by providing an ontological supported evolution and adaptation approach which propagate changes to the DW schema, verify the impact of the change and automatically adjust the dependent entities before implementing the changes at the physical level.
3. To provide optimized horizontal referential partitioning and mixed partitioning to handle DW schema partitioning issues and also adapt the existing partitions in case of evolving queries.

4. To evaluate the proposed approaches with the application to a case study and analyze the obtained results by performing a comparison with existing works.

1.10 RESEARCH CONTRIBUTIONS

To achieve the derived objectives this research work provides different solutions to handle DW schema design, evolution, and partitioning issues. This helps the DW designer or the DW administrator (DWA) to relive from the tedious task of design and management of DW schema and it facilitates achieving it in an efficient and automated way. Following are the three main contributions of the research work:

a) Hybrid approach to automate the multidimensional schema design

- A formal representation of the user requirements and data source is provided through ontology which facilitates automatic reconciliation at the early stage of the design.
- The proposed OntoMD approach provides identification of the multidimensional elements and generation of the DW schema following the conceptual, logical and physical phases of the design process.
- A tool is developed to offer graphical interfaces to the designer to facilitate the use of the proposed hybrid multidimensional modeling approach of a given business domain.

b) Ontological approach to handle evolution and its impact on DW schema

- A formal representation of the user requirements, data source and DW schema is provided through ontology to facilitate the automation of the evolution task.
- A new approach OntoEvol is proposed for the propagation of changes from the requirements and data source to the DW schema at the ontological level.

- The proposed approach provides identification and adaptation of dependent entities that are affected after the changes are propagated to the DW schema.
- A method to perform impact analysis of a change over the DW schema and its dependent modules is provided.

c) Optimized approach to solve DW schema partitioning issues

- A Dimension table selection technique using multiple criteria is provided for referential horizontal partitioning of DW schemas by the proposed ORP approach.
- A formal fragmentation selection approach is developed using hybrid evolutionary algorithms.
- An optimized mixed fragmentation is provided by ORP to solve the big dimension problem in DW schema partitioning.
- The proposed ORP approach includes partition management to apply refragmentation in case of evolving queries.

1.11 ORGANIZATION OF THE THESIS

The thesis is organized into six chapters as given below:

Chapter 1 provides a brief description of the basic concepts of the research work, motivation of the research, problem statement, objectives and research contributions.

Chapter 2 presents the review of related works and comparative study between them. Existing traditional and ontology based approaches for multidimensional schema design are discussed. The evolution approaches such as DW schema evolution and schema versioning are described briefly. A review of existing DW partitioning methods has been provided in this chapter.

Chapter 3 discusses the hybrid multidimensional modeling approach for DW using ontology and the various steps involved. The implementation detail of OntoMD tool which facilitates the automation of the design task is provided. The proposed work applied to a case study is explained in detail. Details of the evaluation applied to the proposed approach are discussed.

Chapter 4 elaborates the development of an ontological approach for handling multidimensional schema evolution. It explains the various steps which use ontology in order to handle the evolution task. A description of the method to handle automatic adaptation after evolution is provided. Evaluations of the proposed evolution approach along with impact analysis are detailed.

Chapter 5 deals with the proposed referential partitioning approach to fragmentation selection problem. It presents the details of dimension selection method and hybrid evolutionary algorithm to optimize horizontal referential partitioning. The proposed optimized mixed fragmentation technique to solve the big dimension problem is elaborated. The partition management used in case of query evolution has been discussed. The evaluation of the proposed partitioning approaches has also been given.

Chapter 6 provides the conclusion by summarizing the research work and suggesting possible future enhancements.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The DW data is modeled multidimensionally to facilitate complex analysis and visualization of business information. Thus, the design of the DW schema requires two important considerations. First, is the design of the DW schema for the warehouse and second one deal with obtaining a schema that satisfies maintenance and performance requirements. Generating the DW schema involves identifying the multidimensional elements, and this schema needs to be maintained when the business rules of an organization evolve. Further, the schema is required to be tuned for performance enhancement before it is implemented in order to optimize the end user queries. This chapter provides a brief idea about the existing works available in the literature on the DW schema design, its evolution and partitioning. A comparative analysis of these works has been provided. It also presents the limitations of the existing works which motivated to take up the research work in this area.

Section 2.2 discusses about the DW schema design approaches. The detail about DW schema evolution approaches has been provided in section 2.3. Section 2.4 provides the review of different evolution approaches available in the literature. The summary about the literature survey has been given in section 2.5.

2.2 DATA WAREHOUSE SCHEMA DESIGN APPROACHES

In this section a review of the existing research work on DW schema design has been provided. DW schema design requires specialized design techniques which consists of conceptual, logical and physical phases (Golfarelli and Rizzi, 1999).

The conceptual design allows having closer ideas about the ways that a user can perceive an application domain (Malinowski and Zimányi, 2006). It aims at deriving an implementation-independent and expressive conceptual schema for the DW, starting from the user requirements and from the structure of the data source. This step is considered as a key that ensures the success of the DW projects since it defines the expressivity of the multidimensional schema (Golfarelli and Rizzi, 2009). The result of this step is a graphical notation which facilitates to the designer and the user for understanding and managing the conceptual schema.

The logical design of the DW serves to define the structures to ensure an efficient access to information. It can be presented as a multidimensional structure that takes as input the conceptual schema representation, information requirements, source systems, and non-functional requirements (Peralta and Ruggia, 2003). The process of logical design involves arranging data into a series of logical relationships called entities and attributes. An entity represents a table and an attribute is a component of an entity that helps define the uniqueness of the entity. The logical design results in a set of entities and attributes corresponding to fact tables and dimension tables, and a model of operational data from the source into subject-oriented information in the target DW schema.

Physical design deals with the effective way of storing and retrieving the data from the DW. During the physical design the logical schema needs to be converted into a description of a physical database structure using proper mapping (Golfarelli and Rizzi, 1998). The physical design involves the creation of the database objects like tables, columns, indexes, primary key, foreign keys, views, sequences, etc. It addresses all the issues specifically related to the suite of tools chosen for implementation such as partitioning, indexing and allocation.

According to Winter and Strauch 2004, the DW schema design approaches mainly follow any of the following methodologies:

- **Demand-driven**: Also known as a requirement-driven or goal-driven approach, focus on determining the end-user requirements to produce a multidimensional schema. Only in the later stages, the output schema is mapped onto the data sources.
- **Supply-driven**: Also known as data-driven approach, start from a detailed analysis of the data sources to determine the multidimensional concepts. End-user requirements are eventually considered in the later stages to filter results obtained.
- **Hybrid**: Hybrid approach combines both frameworks, demand-driven and supply-driven. Mostly, these approaches start with a demand-driven stage to identify facts of interest followed by a supply-driven stage to identify its dimensional concepts.

In literature, earlier approaches called traditional approaches used for DW schema design provided a manual or semi-automatic way of constructing the multidimensional schema through step-by-step guidelines. As the whole design task relied on the knowledge and expertise of the designer, there was a need to provide an automation of the design process. The latter approaches available in the literature tried to provide an automation of the design task using ontology and they are classified as ontological approaches. In the following sections a review of the existing traditional and ontological approaches which follows either demand-driven, supply-driven or hybrid methodology is presented along with a comparison study between them.

2.2.1 Traditional Approaches

Prat *et al.,* (2006) proposed a UML-based DW design method following the three design phases such as conceptual, logical and physical. They use a set of meta-models represented in UML and a set of transformations based on Object Constraint

Language (OCL) at each phase in order to facilitate semi-automation of their design task. In the conceptual design the user requirements are represented in the form of UML class diagram. The second step involves the mapping of the UML conceptual model into a logical multidimensional schema. In the logical design phase the enriched and transformed UML conceptual model is mapped into a logical schema. The physical phase maps the multidimensional schema into a physical database schema, depending on the target OLAP tool.

Song *et al.,* (2007) proposed SAMSTAR method, which is a semi-automated lexical method for generating STAR schemas from an entity-relationship diagram (ERD) by analyzing its semantics as well as the structure. It mainly follows the supply driven approach covering the conceptual and logical design phases.

Giorgini *et al.,* (2008) presented an approach called GRAnD (Goal-oriented Requirement Analysis for DWs). It is a goal-oriented approach to requirement analysis for DWs based on the Tropos methodology (Bresciani *et al.,* 2004). Their approach could be considered as demand driven when only requirements are used to derive the conceptual multidimensional model. It could also be considered as mixed approach when both source and requirements knowledge is used. The GRAnD adopts two different perspectives for requirement analysis: organizational modeling and decisional modeling. The organizational modeling is centered on stakeholders and decisional modeling focuses on the actors. After the requirement analysis the next step derives the conceptual multidimensional model. In GRAnD, facts, dimensions and measures identified during the requirements analysis is mapped manually over the data sources.

Maz´on *et al.,* (2009) proposed a framework based on the Model Driven Architecture (MDA) for the development of a hybrid multidimensional model at the conceptual and logical level. The approach uses the information requirements model (Computation Independent Model, CIM) obtained from decision makers to derive the conceptual multidimensional model of the DW (Platform Independent Model, PIM). PIM is then reconciled with the data sources which are marked with its multidimensional concepts to obtain hybrid PIM. By considering the deployment

platforms, several logical models are derived from this hybrid PIM as Platform Specific Models (PSMs). The relations between models are implemented by using the Query/View/ Transformations (QVT) language. Several transformations using the Model to Text Transformation (Mof2Text) language are defined in order to obtain the code for the implementation of the multidimensional model according to each PSM.

Abelló and Romero (2010) presented an approach for supporting multidimensional design based on Multidimensional Design by Examples (MDBE), which is a semi-automated method to carry demand driven and supply driven in a parallel way. The method uses end-user information requirements (expressed as SQL queries) and the logical model of the data sources as inputs. It produces a constellation schema from the data sources as output.

2.2.2 Ontological Based Approaches

The traditional approaches discussed so far mainly work with relational sources, hence they use set of heuristic to derive multidimensional elements such as facts and dimensions. But today, the information systems are dealing with semi-structured and unstructured sources. These sources may lead to heterogeneous problems which may affect the design output. The three heterogeneity issues that normally arise are syntactic, structural and semantic (Pérez *et al.*, 2008). Resolving syntactic and structural issues may be done irrespective of the context, but semantic issues are application dependent. In order to represent the concept of a domain irrespective of the application, ontology began to be used. Ontology provides a way to represent the concepts of a domain and automatic processing of the concepts. Hence, the use of ontology for the DW design helps to solve the heterogeneity issues that arise in the data sources (Gagnon, 2007) as well as provide automation of the design task (Romero and Abelló, 2010). Following is a brief review of the ontological approaches available in the literature:

Romero and Abelló (2010) presented a method called Automating Multidimensional Design from Ontologies (AMDO). It follows a reengineering

process that derives the multidimensional schema from the data source represented as ontology. The three constraints that are used to identify multidimensional concepts in data source are: multidimensional model constraint, multidimensional space arrangement constraint, summarization integrity constraint.The multidimensional concepts obtained from the data source are filtered according to certain filtering conditions.

Romero *et al.,* (2011) presents a system called GEM (Generating ETL and Multidimensional designs). The system provides designers with a semi-automatic method for generating the conceptual multidimensional schema. It is also used to derive a conceptual representation of the ETL processes that coordinate the data flow from the data sources to the DW system. GEM uses a set of data sources represented as OWL ontology and business requirements expressed in XML format. It carries out the requirement validation by mapping it to the source. An ontology subset is derived for each requirement and the multidimensional elements are tagged for the concepts in the ontology.

Jovanovic *et al.,* (2012) proposed a semi-automatic method called Ontology-based DW REquirement evolution and integration (ORE), for constructing the multidimensional schema. Their approach considers each requirement separately, and incrementally builds the unified multidimensional schema satisfying the entire set of requirements. The information requirements are validated against the available data sources. The source subset satisfying the given requirements is interpreted with the identified multidimensional knowledge. Here, the multidimensional interpretations (MDI) satisfying the given set of information requirements are considered as inputs. The ORE comprises four stages, namely matching facts, matching dimensions, complementing the multidimensional design, and integration. In all stages, they maintain a structure, called traceability metadata (TM), for systematically tracing about the multidimensional design that is integrated at each stage.

Selma *et al.,* (2012) proposed a method, namely the ontology-based DW (OBDW) and case tool for designing a DW multidimensional schema from

ontology-based database sources. They proposed a goal-oriented requirement model based on which DW ontology is extracted from the global ontology representing the sources. The DW ontology is annotated by the multidimensional concepts. The multidimensional role of concepts and properties are identified and annotated over the ontology, based on the analysis of the defined goals. The logical model of the DW is generated by translating the annotated DW ontology.

2.2.3 Comparative Analysis of Schema Design Approaches

In this section a comparison of the traditional and ontological approaches is provided along with a detailed summary of the main features of the design approaches. Most of the features used for comparison represent the existence of the corresponding feature through yes/no and other features have alternative values.

Comparison of Traditional Approaches

Table 2.1 presents the comparison of the traditional approaches. Analyzing Table 2.1 it was found that, most of the approaches considered user requirements to be the important aspect of constructing the multidimensional model. They are classified as demand-driven or goal-driven. Giorgini *et al.,* (2008) and Maz´on *et al.,* (2009) used goal oriented and Prat *et al.,* (2006) used UML diagram for requirements representation. Here, the multidimensional elements such as fact, measure, and dimension are identified from the user requirements. Hence, the main limitation of the demand driven approaches is that, the results obtained depends on the correctness and understanding of the requirements in hand. To handle this issue, few approaches such as Prat *et al.,* (2006) used data source at the end of the design to produce the final model.

Few researchers realized that the knowledge contained in the data source to be helpful for the schema design and its automation. These are called as supply driven or data driven approaches. Relational sources are used by Giorgini *et al.,* (2008) and Abelló and Romero (2010), ER diagram is used by Song *et al.,* (2007) as data source. Most of the approaches use different patterns or heuristics to identify

concepts likely to play a multidimensional element. Hence, the design experts might require a well documented data source, otherwise the final model may be not be derived accurately. Moreover, as user requirements are not considered by most of the supply driven approaches, they produce exhaustive results in the final output, which may not satisfy the end user goals.

Few supply driven approaches such as Song *et al.*, (2007) consider user requirements to filter the final results, but they perform the demand driven stage in a manual way. Hence, the ideal scenario for deriving the DW schema would require a hybrid approach, i.e., a combination of supply-driven and demand-driven stages. Thus, the resulting multidimensional schema would satisfy user requirements and be conciliated with the data sources simultaneously.

In literature only few approaches such as Giorgini *et al.*, (2008) and Abelló and Romero (2010) follow a hybrid methodology and try to automate the design process. But, the degree of automation achieved is rather low. Specifically, these approaches consist of a detailed requirements elicitation stage that need to be performed manually and an automated analysis of the data sources. At the end of the design, both stages are put in common by conciliating the data sources and requirements. In these approaches the requirements elicitation stage leads the process and the main design decisions are made in this step. The analysis of the data sources is carried out only in a superficial manner.

Considering the different phases of design, most of the traditional approaches such as Giorgini *et al.*, (2008) and Abelló and Romero (2010) concentrate on conceptual phase generating the conceptual schema and few carry out the conceptual, logical and physical phases (Prat *et al.*, 2006; Maz´on *et al.*, 2009).

Table 2.1 Comparison of Traditional Approaches

Features	Prat *et al.,* (2006)	Song *et al.,* (2007)	Giorgini *et al.,* (2008)	Maz´on *et al.,* (2009)	Romero and Abelló (2010)
Automation	Semi-automatic	Semi-automatic	Semi-automatic	Semi-automatic	Semi-automatic
Design Approach	Demand-Driven	Supply-Driven	Demand-Driven & Hybrid	Demand-Driven	Hybrid
Requirement Representation	UML diagram	-	i* framework	i* framework	SQL
Data source Representation	-	ER - diagram	Relational	-	Relational
Formal Algorithm	No	Yes	No	No	Yes
Conceptual Design	Yes	Yes	Yes	Yes	Yes
Logical Design	Yes	Yes	No	Yes	No
Physical Design	Yes	No	No	Yes	No
Quality assessment	No	No	No	No	No
User Suggestion	No	Yes	No	No	No
Tool	Yes	No	Yes	No	Yes

Comparison of Ontological Approaches

Table 2.2 provides the comparison of the existing ontological approaches. Like traditional approaches, the ontological approaches follow supply driven, demand driven or hybrid methodology to perform the DW schema design.

Table 2.2 Comparison of Ontology Based Approaches

Features	Romero and Abelló (2010)	Romero *et al.*, (2011)	Jovanovic *et al.*, (2012)	Selma *et al.*, (2012)
Automation	Semi-automatic	Semi-automatic	Semi-automatic	Semi-automatic
Design Approach	Supply-Driven	Demand-Driven	Demand-Driven	Demand-Driven
Requirement Representation	-	XML Format	Natural Language	Ontology
Data source Representation	Ontology	Ontology	Ontology	Ontology
Formal Algorithm	Yes	Yes	No	Yes
Conceptual Design	Yes	Yes	Yes	Yes
Logical Design	No	No	No	Yes
Physical Design	No	No	No	No
Quality assessment	No	Yes	No	No
User Suggestion	Yes	Yes	No	No
Tool	Yes	Yes	No	Yes

The supply driven approach AMDO (Romero and Abelló, 2010) used ontology representation of the data source to automatically derive the multidimensional elements. Here the results produced are exhaustive, and hence some filtering function is required to filter the obtained results. Other ontological approaches such as GEM (Romero *et al.,* 2011) and OBDW (Selma *et al.,* 2012) used both user requirements and data source for the design process. Though these approaches claim to use the hybrid methodology, they are not a pure hybrid as the entire design process is driven by the user requirements. A thorough analysis of data source is lagging in these approaches and moreover the knowledge contained in it is used superficially.

To derive the multidimensional elements, most of these approaches (Romero and Abelló 2010; Romero *et al.,* 2011; Selma *et al.,* 2012) provided a formal algorithm. But considering the design phases, Romero and Abelló (2010), Romero *et al.,* (2011) and Jovanovic *et al.,* (2012) proposed approaches cover only the conceptual phase to produce the conceptual multidimensional schema and Selma *et al.,* (2012) proposed approach covers conceptual and logical phases.

In a real scenario there are multiple heterogeneous sources exists and to utilize the knowledge in them, these sources need to be integrated into a single source schema in order to carry the design task. Except Selma *et al.,* (2012), other approaches do not consider about the source integration. Even with this approach they discussed only about the integration of ontology based databases (OBDBs) and do not consider other type of data sources.

In order to provide a semi-automatic means of deriving the multidimensional schema, the existing approaches managed to develop a prototype tool which would facilitate the designer to perform the design process.

Limitations

1. Compared to demand-driven and supply driven approach, the hybrid approaches provide a promising results for the DW schema design. However, in the existing hybrid approaches the mechanisms through which to formally match the data sources with user requirements in the early stages of the design are not investigated so far.
2. In the hybrid methodology, the identification of multidimensional elements (i.e., fact, dimensions, dimension hierarchy) in the data sources is a mandatory previous step before reconciling requirements and data sources. These elements are usually annotated in a manual (Giorgini *et al.,* 2008) or semiautomatic (Mazón and Trujillo, 2009; Song *et al.,* 2007) manner from the data sources using syntactic information (Lechtenbörger and Vossen, 2003) which is not enough for every scenario and prevents their total automatization.

3. The quality of the schema designed need to be analyzed before it is carried out for actual implementation. The existing approaches do not focus on ensuring the schema quality in a formal way.
4. Though existing approaches provide tools to carry out the schema design, they do not provide the required functionalities to fully automate the design task.

Thus, in this research the proposed DW schema design approach considers both the user requirements and data source at the early stage of the design. An appropriate multidimensional model is generated by covering different phases of the design in an automated way.

2.3 DATA WAREHOUSE SCHEMA EVOLUTION APPROACHES

Evolution in DW may be generated by change in data source schema and in requirements. Following are the various reasons for changes happening in the data source or user requirements:

1. Incorporation of new users or requirement in the system or creating new versions (Rechy-Ramirez and Benitez-Guerrero, 2006).
2. Ambiguous or insufficient requirements during the development phase (Body *et al.,* 2002).
3. Periodical revisions done for the removal of bugs and redundancies (Bebel *et al.,* 2006).
4. Change in the requirements during the operational phase of the DW which results in the structural evolution of the DW schema. (Rechy-Ramirez and Benitez-Guerrero, 2006)
5. Reorganization of the DW schema during the operational phase of the DW as a result of different design solutions that are decided upon. (Bebel *et al.,* 2006).
6. Change in the information source resulting in a new DW design.

The DW must evolve in reaction to the above mentioned reasons. Data changes are monitored and propagated to a DW often by means of materialized views (Chen *et al.,* 2006; Sahpaski *et al*., 2009) and the history of data changes is supported by applying temporal extensions (Eder *et al,*. 2002; Eder *et al.,* 2006). Whereas, source schema changes are often handled by applying schema evolution (Benitez-Guerrero *et al.,* 2004; Curino *et al.,* 2009 ; Fan and Poulovassilis, 2004) and schema versioning techniques (Bebel *et al.,* 2006; Papastefanatos *et al.,* 2007; Rechy-Ramirez and Benitez-Guerrero, 2006 ; Sahpaski *et al.,* 2009). In schema evolution approaches historical DW states are lost as there is only one DW schema that is being modified. In schema versioning approaches only historical versions of data are maintained, whereas schema modifications are difficult to handle. Thus, the DW evolution is classified into two main approaches, namely schema evolution and schema versioning. In this section a brief review of these approaches has been provided.

2.3.1 Schema Evolution Approaches

Benitez-Guerrero *et al.,* (2004) proposed a Warehouse Evolution System (WHES) that demonstrates a DW evolution model and its associated multidimensional data definition language. WHES implements a set of translation rules to provide one-to-one mapping between multidimensional schema and the relational model. It also defines a set of propagation rules that modify the relational model whenever a change occurs in the corresponding multidimensional model. The authors have proposed 16 operators to modify the multidimensional schemas.

Papastefanatos *et al.,* (2007) provides a framework for performing ETL evolution for potential changes to data source of a DW. They use a graph model that uniformly models relations, views, queries, ETL operations, and their significant properties. The authors propose a set of rules to annotate the graph representing the ETL workflow. The annotation contains actions that either blocks the event or reshapes the graphs when a change event occurs. The change event along with the annotation represents policy to be followed for the handling of a potential change.

Banerjee *et al.*, (2009) contributes formalism for representing DW schemas and determining the validity of schema evolution operators applied to a schema. The schema evolution operators are the core features of a DW that are defined in the generalized model. Here, the authors have summarized various schema evolution operators based on change in dimension, change in fact and change to a cube. Along with schema evolution the authors have proposed modeling extended hierarchy semantics.

Solodovnikova and Niedrite (2011) proposed a framework DWE to support DW evolution. It allows propagating different changes in DW, creating versions of schemata and data semantics. This approach is user-centric, where users are involved to design reports on multiple DW versions using user terms. The operations of the framework are based on the metadata which is used to describe the DW schema versions and to accumulate information about reports defined by users on schema versions.

Thakur and Gosain (2011) present a theoretical framework called DWEVOLVE to support DW evolution. The changes in the requirements specified by the stakeholders as well as the developers are analyzed here. These changes are then incorporated into the warehouse by performing appropriate additions, deletions and updates. It consists of a module that cleans redundant or dirty data by employing certain cleaning algorithms. To enhance the framework operation comprehensive metadata support has been provided.

2.3.2 Schema Versioning Approaches

Rechy-Ramirez and Benitez-Guerrero (2006) proposed a Version-based Evolution Model based on the bi-temporal schema versioning. In the model, the granularity of versioning represents the multidimensional database version. Here each version is formed by a multidimensional schema and a multidimensional database which is conformed to the schema. Thus, when a change is made to a multidimensional schema, a new multidimensional database version having a new associated temporal pertinence is created. In addition to the evolution operators, the

authors have defined a SQL-like language. This allows the DW administrator to express his/her evolution requirements.

Golfarelli *et al.,* (2006) have proposed an approach to schema versioning and formulating cross-version queries for DW. Their approach allows queries that cover data across different schema versions. The authors have introduced a representation of DW schemata as graphs of simple functional dependencies. They define algebra of schema graph modification operations in order to create new schema versions. The also discuss how augmented schemata can be introduced to speed up the cross-version querying process.

Sahpaski *et al.,* (2009) presented an approach for dynamically evolving the design of the DW schema spanning across its multiple versions. An optimization of the DW implementation schema design is provided by the authors by defining the multiversion data cube and the multiversion implementation schema. They add the multiversion implementation schema with its instances to the generalized solution space of the optimization problem. They also introduce a new derivation procedure and a new derivation cost constraint. The procedure is used for evolving the current implementation schema to a new implementation.

Oueslati and Akaichi *et al.,* (2011) proposed an approach called Multiversion Trajectory Data Warehouse (MVTDW). The main goal of this approach is to propose a solution, based on versioning approach which is able to handle structural changes in order to keep track of the DW evolution. There are two types of schema versioning used in this approach: real version and alternative version. The real version is defined as a version that handles changes of the real world like changing geographical borders of countries. The alternative version is defined as a version that handles virtual business scenarios. The authors also proposed certain constraints that have to be fulfilled to guarantee the integrity of the MVTDW.

Xuan *et al.,* (2006) presented a solution for asynchronous versioning problem for an ontology-based DW. This DW integrates ontology-based data sources which are autonomous and heterogeneous. The data source contains local

ontologies which references a shared ontology by subsumption relationships. These local ontologies change their schema as well as instances with respect to changes in the environment. To manage ontology changes the authors proposed two different solutions: ontology evolution and ontology revolution.

2.3.3 Comparitive Analysis of Schema Evolution Approaches

Table 2.3 provides the comparison of various DW evolution approaches. The schema evolution approaches focus on updating the changes over the original schema. Whereas the schema versioning approaches maintains the old schema and creates the new version of the schema by updating the changes.

Table 2.3 Comparison of Evolution Approaches

Existing Methods	Approach	DW Schema	Source Change	Requirement Change	Evolution Operators	Formal Method	Automatic Adaptation	Impact Prediction
Benitez-Guerrero *et al.,* (2004)	Schema Evolution	Relational	Yes	No	Yes	MDL	No	No
Papastefanatos *et al.,* (2007)	ETL Evolution	Graph	Yes	No	No	Algorithm	Yes	Yes
Banerjee *et al.,* (2009)	Schema Evolution	Relational	Yes	No	Yes	Algorithm	No	No
Thakur and Gosain (2011)	Schema Evolution	Relational	Yes	Yes	No	-	No	No
Solodovnikova and Niedrite (2011)	Schema Evolution & Schema Version	Logical & Relational	Yes	Yes	No	-	Yes	No
Rechy-Ramirez and Benitez-Guerrero (2006)	Schema Version	Conceptual	Yes	No	Yes	SQL-Like	No	No
Golfarelli *et al.,* (2006)	Schema Version	Graph	No	Yes	No	Schema Modification Algebra	No	No
Sahpaski *et al.,* (2009)	Schema Version	Relational	Yes	No	No	-	No	No
Oueslati and Akaichi *et al.,* (2011)	Schema Version	Relational	Yes	No	Yes	Algorithm	Yes	No
Xuan *et al.,* 2006	Schema Version	Ontology	Yes	No	No	-	No	No

DW schema needs to evolve when business requirements are changed or extended or a data source schema is adapted after changes. Most of the existing work concentrates on handling source schema evolution and only few approaches (Thakur and Gosain, 2011, Golfarelli *et al.,* 2006) discuss about handling requirement changes.

The three types of changes as given in the literature, that normally occur over the DW schema are addition, deletion and rename. These changes are carried over the multidimensional elements such as fact, fact attributes, measures, dimension, dimension attributes, level, level attributes and fact-dimension relationship. In order to perform these changes, different set of evolution operators are proposed by several authors (Benitez-Guerrero *et al.,* 2004; Rechy-Ramirez; Banerjee *et al.,* 2009; Oueslati and Akaichi *et al.,* 2011). Apart from evolution operators certain existing approaches (Benitez-Guerrero *et al.,* (2004); Rechy-Ramirez and Benitez-Guerrero (2006); Golfarelli *et al.,* (2006); Papastefanatos *et al.,* (2007); Banerjee *et al.,* (2009); Oueslati and Akaichi (2011)) provide formal methods such as algorithms and SQL like languages to update the changes over the DW schema.

The existing approaches propagate the changes directly over the relational schema of the DW except the works by Papastefanatos *et al.,* (2007) and Golfarelli *et al.,* (2006), which handle the evolution over the graph representation of the multidimensional schema. The change propagation is handled manually by these works. Whereas, few works (Papastefanatos *et al.,* (2007), Solodovnikova and Niedrite (2011) and Oueslati and Akaichi *et al.,* (2011)) tried to automate the evolution task.

The DW schema is not an independent entity. When it evolves, it might have an impact on its dependent modules such as ETL tasks, queries, views, etc. But none of the existing approaches concentrate on the impact analysis and the cost of handling changes over the DW schema.

Limitations

1. Most of the existing approaches either schema evolution or schema versioning handled changes in the data source schema. Very few concentrated on changes in business requirements, but they do not provide a concrete way of handling these changes.
2. Evolution operators and formal methods such as algorithms and SQL-like language are proposed by different authors to propagate changes over the DW. Still, most of the approaches could not provide an automation of the evolution task.
3. The changes are mostly updated over the relational schema of the DW. Hence, such approaches might not produce a feasible solution to the evolution process as they may incur high maintenance cost. Only a few works over the conceptual or graph representation of the DW schema, but they are found to provide a complex scenario in order to handle the evolution.
4. From the existing works, it is observed that none of the approaches discussed about the impact that the updated DW schema might bring to its dependent entities. Papastefanatos *et al.*, 2007 focus on impact analysis, but their main concentration is on ETL evolution rather than DW schema evolution.
5. Finally, there is a lack of automated approach which might help the DW designer to carry out the complex process of evolution in an efficient way.

The problem of managing changes on the schema level, which may be demanded by changes either in the user requirements or in the sources has been addressed by this research. An automated approach has been proposed to propagate changes from user requirements or data source over the DW schema along with adaptation of the dependent entities.

2.4 DATA WAREHOUSE SCHEMA PARTITIONING APPROACHES

The DW schema needs to be partitioned for optimization of its performance before it is implemented in the underlying database. Partitioning is the process of splitting large relations (tables) into smaller ones so that the database needs to retrieve only relevant data at a particular time (Bellatreche *et al.,* 2000; Papadomanolakis and Ailamaki, 2004). The two ways to partition a relation are: horizontal and vertical (Sanjay *et al.,* 2004). The horizontal partitioning involves splitting the tuples (rows) of a relation and placing them into two or more relations with the identical structure. Vertical partitioning involves splitting the attributes (columns) of a relation and placing them into two or more relations linked by the relation's primary key. The main advantages of partitioning are: it can significantly impact the performance of the workload, i.e. the set of queries that executes against the DW system by reducing the cost of accessing and processing data. Moreover, it allows parallel processing of data by locating tuples where they are most frequently accessed (Bellatreche *et al.,* 2000).

Different partitioning modes have been proposed and supported by various database systems. They are classified into two main types, based on the number of participating tables in the partitioning process: single table partitioning and table dependent partitioning. In case of single table partitioning, a table is partitioned based only on its attributes. Thus, the partitioning is similar to primary horizontal partitioning (Ozsu and Valduriez, 1999). In order to implement this partitioning, several modes exist: Range, List, Hash, Round Robin (supported by Sybase), Composite (Range-Range, Range- List, List-List, etc.) and Virtual Column partitioning. The single table partitioning is well adapted for optimizing selection operations, especially when partitioning key matches with selection attributes. In case of table dependent partitioning, a table (usually called child) is partitioned based on the fragmentation schemes of other tables (called parents). This type of partitioning is feasible if there is a parent-child relationship between these two tables (Eadon *et al.,* 2008). To implement this partitioning, the two main modes that are

available are: Native Referential Partitioning and User-driven Referential Partitioning.

In the native referential partitioning, a child table inherits the partitioning characteristics of its parent table. It is supported by Oracle11G, by the use of a native DDL. It optimizes selection and joins simultaneously. Hence, it is well adapted for star join queries. User-driven referential partitioning is a manual implementation of referential partitioning, i.e., it is implemented using the single table partitioning. That is, a parent table is first horizontally partitioned on its primary key, then the child table is split using the foreign key referencing the parent table. This kind of partitioning has been used for designing parallel DWs and is not well adapted for star join queries.

Thus, the partitioning approaches available in the literature can be classified as a single table and referential partitioning. Single table partitioning, involves fragmenting the fact table. Referential partitioning, involves fragmenting the fact table with reference to fragments of dimension table(s). This section presents the review of these existing partitioning approaches. When the DW table(s) is partitioned, it generates large set of partitions or fragments. Hence selection of optimal fragment has been the focus of several approaches that are discussed here.

2.4.1 Single Table Partitioning Approaches

Brkić *et al.,* (2012) discussed the procedure for horizontal fragmentation of DW tables. They argue that their procedure is suitable when there are multiple independent organizational units that produce data of the same structure and which are needed to be loaded into the consolidated fact table. In such case, in order to ensure data quality factors such as completeness and timeliness, their proposed method uses meta-data for horizontal fragmentation of the DW. They define certain expression for completeness and timeliness of DW tables which is used to ensure the quality of the DW after ETL operation. They also provide experimental results to show how horizontal partitioning of the fact tables can improve the quality of the warehouse when compared to the non partitioned warehouse.

Liu and Iftikhar (2013) presented the design method for modeling big dimensions in a DW. In order to automate the DW modeling process they use OWL ontology to describe the semantics of a big dimension. They consider the vertical, horizontal and hybrid partitioning technologies for modeling big dimension. Their approach streamlines the modeling process from conceptual to physical DW design.

Barr (2013) presented a work that deals with the problem of selecting the horizontal fragmentation. It considers two objective functions to minimize that is, the number of I/O between memory and disk during decisional queries and the number of fragments while selecting the fragment. A scalar method called compromise method has been used. It is responsible to optimize an objective function which considers the second objective function as a constraint. The main principle of the method is to transform a multi-objective problem into single objective one under additional constraints. To realize the meaning of compromise of the proposed multi-objective optimization method, here the results obtained using Genetic Algorithm are collected. These results give the number of inputs/outputs according to the number of fragments introduced. Thus, the horizontal fragments that minimize both the number of I/O between memory and disk and number of fragments are selected.

2.4.2 Referential Partitioning Approaches

Boukhalfa *et al.,* (2009) proposed an architecture for the combined selection of horizontal partitioning (HP) and bitmap join indexes (BJI) by exploiting their similarities. Here the dimension table is partitioned using several attributes without their restriction to be contained in a hierarchy. The fact table is partitioned based on the partitioning of the dimension table. For a fragmentation schema generation, they used genetic algorithm. To generate a configuration of BJI they used greedy algorithm.

Mahboubi and Darmont (2009) proposed a technique which adapts derived horizontal fragmentation techniques developed for relational DWs to the XML DW. In their fragmentation methodology, first the primary horizontal fragmentation is

applied onto the warehouse dimensions using either the predicate construction method or the affinity-based method. Both these methods, inputs selection predicates from the query workload. The facts are finally fragmented according to horizontal fragments obtained by applying either the predicate construction or affinity-based method on dimensions. Fragmentation of facts is achieved by semi-join operations based on a virtual key reference.

Dimovski *et al.*, (2011) proposed a formal approach for horizontal partitioning and its application for optimizing DW design in a cost-based method. The horizontal partitioning is based on predicate abstraction which maps the domain of a relation to be partitioned to an abstract domain following a finite set of arbitrary predicates chosen over the whole concrete domain. In order to address the optimization problem a minimal set of predicates for each relation is derived using ComputeMin procedure. The chosen predicates are used to horizontally partition some (or all) dimension relations of the DW with star schema. The fact relation is partitioned by using the predicates specified on dimension relations. Finally, they use genetic algorithms, known evolutionary heuristic, to find a suitable partition which minimizes the query cost.

Bellatreche (2012) proposed a comprehensive procedure for referential partitioning in the DW. First, it selects relevant dimension table(s) to partition the fact table and they are associated with selection predicates. Each dimension table is then partitioned using single table partitioning type. To generate partitioning scheme of the chosen dimension tables, DBA chooses the hill climbing algorithm. Finally, the fact table is partitioned using referential partitioning based on the fragmentation schemes generated by the algorithm.

2.4.3 Comparitive Analysis of Schema Partitioning Approaches

Table 2.4 provides the comparison of different partitioning approaches available for DW. These approaches aim to minimize the query execution time by applying partitioning over the DW tables there by optimizing its performance. Horizontal partitioning has been employed by most of the approaches. Boukhalfa *et*

al., (2009), Brkić *et al.,* (2012), Liu and Iftikhar, (2013) and Barr (2013) applied single table partitioning strategy in order to partition the fact table. Whereas, Mahboubi and Darmont (2009) and Bellatreche (2012) followed referential partitioning strategy, where the fact table is fragmented based on the fragmentation of dimension table(s). They argue that referential partitioning improves query performance as DW involves star join queries that related fact with the dimension table.

Table 2.4 Comparison of Partitioning Approaches

Existing Methods	Approach	Partitioning	Dimension Selection	Attribute / Predicate Selection	Fragment Selection	Optimization Algorithm	Automation	Tool Support
Brkić *et al.,* 2012	Single	Horizontal	No	No	No	No	No	No
Liu and Iftikhar, 2013	Single	Horizontal Vertical Hybrid	No	No	No	No	Yes	Yes
Barr 2013	Single	Horizontal	No	No	Yes	Genetic and Pareto Dominance	No	No
Boukhalfa *et al.,* 2009	Referential	Horizontal	No	No	Yes	Genetic	No	No
Mahboubi and Darmont 2009	Referential	Horizontal	No	Yes	No	No-	No	No
Dimovski *et al.,* 2011	Referential	Horizontal	No	Yes	Yes	Genetic	No	No
Bellatreche 2012	Referential	Horizontal	Yes	No	Yes	Hill Climbing	Yes	Yes

When partitioning is applied over the dimension or fact table the number of fragments produced is very large depending on the chosen partitioning attributes. Hence, the existing approaches such as Boukhalfa *et al.,* (2009), Dimovski *et al.,*

(2011), Barr (2013) and Bellatreche (2012) used fragmentation selection algorithms such as genetic and hill climbing algorithms to choose optimal fragments. The selection of partitioning attributes is addressed by Mahboubi and Darmont (2009) and Dimovski *et al.,* (2011).

The DW involves multiple dimension tables, hence the choice of dimension table affects the partitioning process. Bellatreche (2012) discussed about dimension table selection method. Also, Bellatreche (2012) and Liu and Iftikhar (2013) provided steps for automating their proposed method along with a prototype tool to facilitate the partitioning process.

Limitations

1. None of the existing approaches focus on dimension table selection except Bellatreche (2012). Hence, as the DW involves multiple dimension tables an efficient selection method need to be developed.
2. Another important issue that needs attention is partitioning attribute selection, which needs to be improved.
3. Most of the approaches follow horizontal partitioning assuming the dimension tables to have a large set of records. But in reality, these tables may contain a large set of attributes. Hence, a mixed partitioning i.e. horizontal and vertical partitioning needs to be combined to fragment a big dimension. Liu and Iftikhar (2013) discuss about traditional mixed partitioning without any partitioning selection procedure involved.
4. The existing approaches, though provide a fragmentation selection method and fragmentation selection algorithm, they do not provide an optimized result.
5. Finally, as the queries imposed over the DW evolves, the existing partitions need to be altered, which has not been discussed in the existing methods.

This research solves the partitioning issues of DW schema by providing optimization of the traditional partitioning techniques. It also enables the partitions to be managed in case of query evolution.

2.5 SUMMARY

In this chapter the different approaches that exist in the DW schema design, evolution and partitioning have been discussed. The existing schema design approaches do not provide a formal mechanism to match the data sources with user requirements. Further, a design method covering the conceptual, logical and physical phases is required to produce the final model. The research works on DW evolution handles data source changes at the physical level, which incurs high maintenance cost. Manual adaptation of DW evolution needs a lot of effort and rework, hence automatic adaptation of DW schema changes and its dependent modules is essential. Related to DW schema partitioning the existing works proposed algorithms for optimal fragment selection. Problems related to dimension and attribute selection, fragmentation of big dimension and partition management has not been much studied.

A detailed comparison of the existing approaches and their limitations are presented in this chapter. From the survey, it is inferred that the DW project to be successful, it is essential to provide an appropriate multidimensional schema which facilitate to make critical business decisions. Thus, this research work focuses on the issues existing in the DW schema design and its management. A comprehensive approach has been developed in order to handle the problems that exist in the DW schema design, evolution and partitioning.

CHAPTER 3

ONTOLOGY BASED APPROACH FOR DATA WAREHOUSE SCHEMA DESIGN

3.1 INTRODUCTION

The DW, owned by an organization, integrates data from heterogeneous sources to enable better decision making. To facilitate OLAP queries the data within the DW is arranged in a multidimensional format. The design of the multidimensional schema in literature has been carried out from requirements or data source and is called as demand driven or supply driven approach, respectively. As discussed in the previous chapter, the design of multidimensional schema requires a hybrid methodology, utilizing the knowledge contained both in requirements and the data source. Hence, in order to ascertain that source system and the requirements are well understood, it is of great importance to provide a formal, explicit, and well-defined way to represent these entities. To facilitate such representation and to provide the automation of multidimensional design the DW community has adopted ontology (Pardillo and Mazón, 2011; Romero and Abelló, 2010; Selma *et al.*, 2012). The existing ontology-based approaches do not fully utilize the benefits of hybrid methodology of the schema design. Moreover, these approaches do not cover the different phases of DW design such as conceptual, logical, and physical. Hence, to overcome the limitations of existing methods, the proposed work has the following contributions:

1. Representation of business requirements in a formal way using ontology.
2. A thorough analysis of the data source to obtain the multidimensional knowledge it contains.
3. Reconciliation of data source and requirements concepts to generate conceptual model.
4. Generation of a logical model using schema transformation and design rules.
5. Physical implementation of the generated schema to validate the quality of the output.

Section 3.1 describes in detail about the proposed OntoMD approach for DW schema design. The OntoMD tool developed based on the OntoMd approach is explained in section 3.2. Section 3.3 provides the details of the case study used for the application of OntoMD approach. The evaluation of the proposed approach is given in section 3.5. The summary of this chapter is provided in section 3.6.

3.2 OntoMD: PROPOSED ONTOLOGY BASED MULTIDIMENSIONAL SCHEMA DESIGN APPROACH

This section describes the steps involved in the proposed ontology based multidimensional (OntoMD) schema design approach. Figure 3.1 represents the steps involved in the proposed approach. A hybrid methodology is followed here for identifying the multidimensional concepts such as facts, measures, dimensions and levels in order to construct the DW schema. To provide automation of the design task, the requirements and the data source are represented in OWL ontology format. Here, the data source is first analyzed to derive the multidimensional concepts it contains. Next, the requirements are matched with the identified multidimensional concepts to filter the required elements of the conceptual schema. From the derived conceptual schema, the

logical schema is generated automatically. Finally, the logical schema is refined using queries to facilitate the physical implementation.

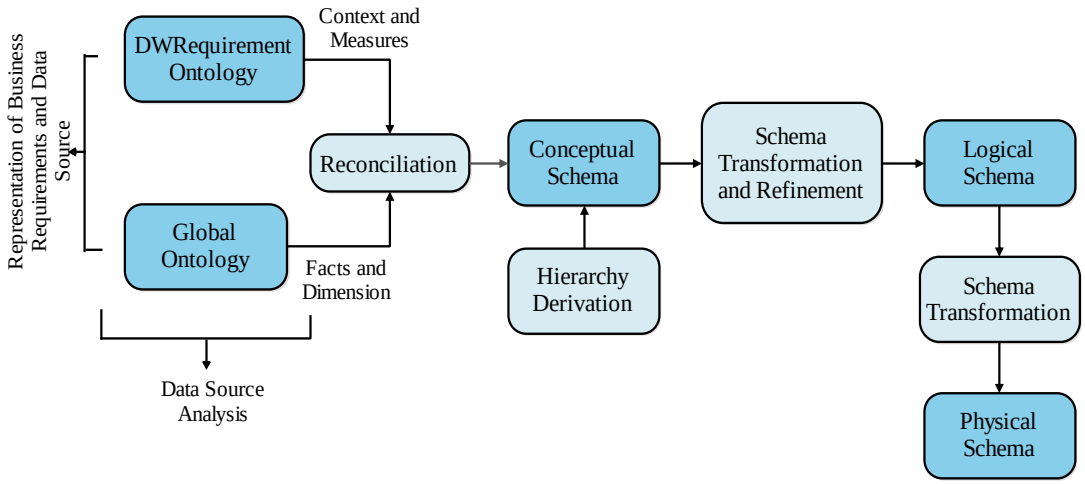

Figure 3.1 Proposed OntoMD Approach

3.2.1 Representation of Business Requirements and Data Source

Like any software system, requirements play a major role in the DW design. Among the several approaches that exist for DW requirement analysis, the goal-oriented approach based on i* framework is found to be an efficient method (Gloria *et al.*, 2008; Inmon, 2005; Niedrite *et al.*, 2007). The proposed OntoMD approach assumes that a requirement analysis has been carried out earlier and the corresponding requirement specification is available for the given business domain. For automation of the design task, it is required to formalize the requirements specification. At this stage, the usage of ontology is beneficial, and thus the requirement ontology is constructed based on the i* framework for DW (Gloria *et al.*, 2008). This ontology captures the multidimensional elements of the business domain through which the knowledge contained in the data source is retrieved.

Figure 3.2 represents the graphical view of the concepts available in the ontology. The DW requirements are represented in the form of goals. Three types of goals that exist in the ontology are *strategic goal*, *decision goal* and *information goal*. The *strategic goals* (for example, ‘‘increase sales”) are the main objectives of the business process. It has one or more *decision goals* (for example, ‘‘determine some kind of promotion”), which determines how a *strategic goal* can be achieved. The *decision goal* in turn has set of *information goals* (for example, ‘‘analyze customer purchases”) to be satisfied. The *information goal* is used to represent sets of *information requirements* of the DW. Each of the information *requirements* contains resources which are defined by three multidimensional elements such as *business process* (the business process to be analyzed), *measures* (process measures under analysis) and *context* (context of analysis) of the business domain.

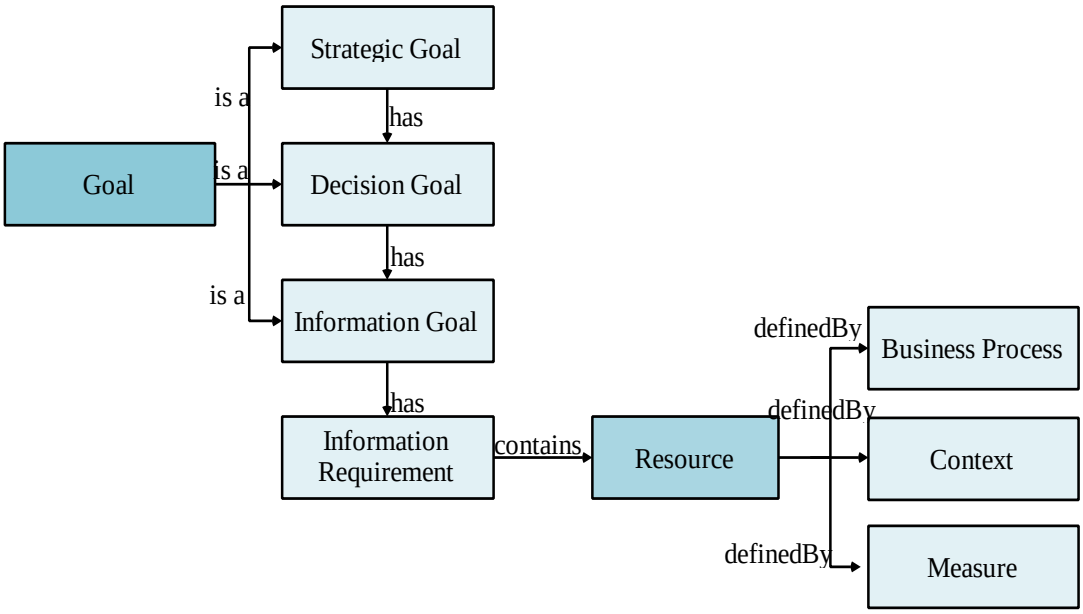

Figure 3.2 Data Warehouse Requirement Ontology

Formally, the DW requirement ontology (DWRO) can be defined as:

DWRO = {*S, I, D, IR, BP, M, CN*}, where,

- *S* is a set of OWL classes representing the strategic goals;
- *I* is a set of OWL classes representing the information goals;
- *D* is a set of OWL classes representing the decision goals;
- *IR* is a set of OWL classes representing the information requirements;
- *BP* is a set of OWL classes representing business process;
- *ME* is a set of OWL classes representing measures;
- *CN* is a set of OWL classes representing the contexts.

The main advantage of the proposed requirement ontology is that when there are any changes in the requirements during the design or at later stages, these changes can be easily incorporated into the ontology. And thus, it facilitates for the evolution of DW which has been discussed in the next chapter.

Once the requirements are formalized, the next step is to represent the data source in an OWL ontology format and perform the required analysis to derive the multidimensional concepts. As the data for the DW are derived from several sources (e.g. Relational, XML, Text) each may have different syntax, semantic, and structural representations. Hence, each source needs to be represented in an ontology format. In order to obtain a unified view of these sources, the ontologies representing them are mapped and merged (Noy and Musen, 2003) to form an integrated ontology. The process of obtaining the integrated data source ontology is given in Figure 3.3.

Different tools are available for converting a particular type of source to OWL representation. A relational source can be converted to ontology using DB2OWL (Cullo *et al.,* 2007) tool. Similarly xml and text sources can be converted to ontology representation using XML2OWL (Lacoste *et al.,* 2011) and OntoLT (Buitelaar *et al.,* 2004) tools respectively. This integrated ontology is taken as input to derive the DW schema.

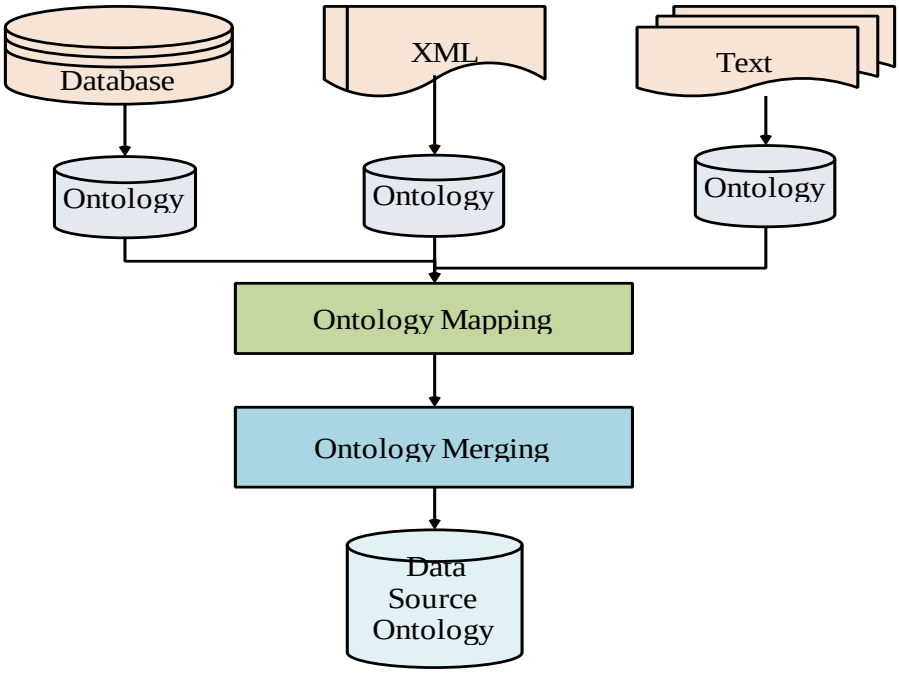

Figure 3.3 Ontology Integration

The integrated data source is described by an ontology called as data source ontology (DSO). DSO is the collection of classes, its data type properties, and object properties which is defined as follows:

DSO = *{C, DTP, OTP}* where,

- *C* is a set of OWL classes;
- *DP* is a set of data type properties;
- *OP* is a set of object type properties.

For any $dpi \in DP$, there exists a domain $c_i = D(dp_i)$ where $c_i \in C$, and a range $c_j = Rng(dp_i) \in DTxml$ where *DT* is the collection of XML Schema data types. For any $opi \in OP$, there exists domain $c_i = D(op_i)$ and range $c_j = Rng(op_i)$ where, $c_i, c_j \in C$, and $i \neq j$.

3.2.2 Analysis of the Data Source

To derive the multidimensional concepts (MDC) from the DSO, the condition of $C \neq \varnothing$ is required. The MDC is represented as follows:

MDC= *{FL, ML, DL}* where,

- *FL* is a set of facts;
- *ML* is a set of measures;
- *DL* is a set of dimensions.

The ontology DSO is taken as input and the MDC are automatically derived through source analysis. From ontology DSO, a concept is identified as fact if it contains enough number of numerical properties. Hence, the concepts with a ratio of numerical attributes greater than the specified threshold are derived as fact. The numerical attributes become the measures of the fact through which a business process is measured. For each fact identified the dimensions are derived by making use of class subsumption and multidimensionality principle which is defined as: *N elements of fact are related to at-least and at-most one element of a dimension through a relationship i.e. an object property*.

In order to derive the MDC contained in the source, a formal algorithm has been proposed. The pseudo code of the algorithm FactDim is given in Figure 3.4. The various notations used in this algorithm have been explained in section 3.2.1. The algorithm first derives the data properties of each class in the ontology. From the available set of data properties the numerical data properties are derived using its range value. The tn and tp represents total numeric data properties and total data properties respectively, for a class (Steps 2 – 8).

```
Input : Ontology of the form O = {C,DP,OP}
Output : MDC = {FL,ML,DL}
1 for all classes ci ∈ C do
2       for all data properties ci .dpi ∈ DP do
3               rng := Rng(ci .dpi), rng ∈ DTxml
4               if isnumeric(rng) then
5                       Add ci .dpi to MLi;
6                       tn++;
7               end if
8               tp++;
9       end for
10      rnp =tn/tp;
11      if (rnp > =threshold) then
12              F= ci;
13              Add ci to FLi;
14              Print(F, ML);
15              Compute_dimension(ci);
16      end if
17 end for
18 Compute_dimension(ci)
19      for all object properties ci.opi ∈ OP do
20              cj:= Rng(ci.opi), ci ∈ C
21              if (cj!= null) then
22                      if (ci.opi allValuesFrom cj &&
                        maxCardinality = = 1) then
23                              Add cj to DLi;
24                      end if
25              endif
26      end for
27      Print(DLi);
```

Figure 3.4 FactDim Algorithm

Next, any ontology class having a ratio of numerical data properties (rnp) greater than a given threshold is considered to be a fact concept. This ratio is obtained by dividing tn by tp (Steps 10 – 13). The numerical data properties of the obtained fact concept are derived as measures. Fact and measures are displayed to the user (Step 14). From the obtained fact concept the dimensions are derived using MDC. For all object properties of the fact class, the algorithm finds the range class. Range class having 1:n relationship with the fact class is derived as dimension concept. Each fact concept may have one or more dimensions (Steps 18 – 27). The facts, measures, and dimension list are then displayed to the user.

3.2.3 Reconciliation of Data Source and Requirements

This step involves in filtering the results obtained from data source analysis using the requirements. The reason for reconciliation is that not all the MDC obtained from the source is useful for business analysis. Hence, the requirement knowledge is used to retrieve only the required concepts. The DWRO represented in Figure 3.2 contains context and measures required for the business analysis. For each information requirement, the context and measure are matched with the obtained list of MDC, and the matched concepts are filtered accordingly. WordNet algorithm (Howe, 2009) is used in order to resolve the syntactic and semantic conflicts during the matching process. Following are the steps in the matching process:

1. For each information requirement from requirements ontology, the contexts and measures are retrieved.
    i. A matching is performed using WorldNet matcher for the contexts of DWO(CN), with the dimensions of MDC(DL) obtained from the FactDim Algorithm
    ii. A matching is performed using WorldNet matcher for measures of DWO(M), with the measures of MDC(ML) obtained from the FactDim Algorithm.
    iii. For the successful matches, the corresponding MDC are retrieved.
    iv. For any unsuccessful match the designer is informed and the corresponding requirement is resolved for any ambiguity.
    v. The matched results containing fact, measures, and dimensions are displayed to the user as a conceptual schema.

Thus, the reconciliation step helps to gather the exact knowledge about the business domain from requirements and source at the early stage of design.

3.2.4 Derivation of Dimension Hierarchy

In order to perform OLAP analysis, such as drill-down or roll-up operations, it is essential to store different level of details within the DW. This step involves finding out the granularity required for representing the dimensions. Having different levels provides a deeper level of business analysis. From the results obtained after reconciliation, the dimension hierarchy containing the levels is computed for each dimension concept.

```
Input : Ontology of the form O = {C,DP,OP}
Output : Dimension Hierarchy: DH

1 Compute_Hierarchy(DL_i)
2     for each class c_i ∈ DL_i do
3         for all object properties c_i.op_i ∈ OP do
4             c_j := Rng(c_i.op_i), c_i ∈ C
5             if(c_j != null && c_j != FL_i )then
6                 if (c_i.op_i allValuesFrom c_j && maxCardinality = = 1) ||
                      (c_i.op_i someValueFrom c_j && maxCardinality = = 1)
                  then
7                     Add c_j to DH_i;
8                 end if
9             end if
10        end for
11        Print(DH_i);
12        Compute_Hierarchy(DH_i)
13    end foreach
```

Figure 3.5 DimHierarchy Algorithm

The proposed DimHierarchy algorithm is used by OntoMD approach to derive the dimension hierarchy. The steps of the algorithm are represented in Figure 3.5. The various notations used in this algorithm have been explained in section 3.2.1. DimHierarchy algorithm uses the input ontology DSO and dimension list *DL* of MDC to compute the dimension hierarchy. Here, for each dimension class, the dimension levels are computed. For all object properties of the dimension class, the algorithm finds the range class (Steps 3-4). Range class having 1:n relationship or

1:1 relationship with the dimension class is derived as level (Steps 5-9). Each level is recursively traversed to compute the dimension hierarchy.

The conceptual model containing fact, measure, dimension, and corresponding dimension hierarchy is presented to the designer for further improvement. Based on the designer's choice the number of levels of a hierarchy is retained.

3.2.5 Generation of Logical Schema

The logical design involves in defining the multidimensional structure for the DW. Thus, in this step of the OntoMD approach the logical schema is derived from the conceptual model. Here, the logical schema is represented in ontology format as given in Figure 3.6. The following transformation rules are followed to construct the DW Ontology (DWO) representing the logical schema:

1. The concept fact is represented as a fact class.
2. Measure concepts become measurable properties of the fact class.
3. The attribute that relate the fact to dimension concepts becomes the object property.
4. The attributes of fact concept are represented as data properties of the fact class.
5. The dimension concepts are represented as separate dimension class.
6. The attributes of each dimension concept become the data properties of the dimension class.
7. For each level concept, a separate class is created with the properties.

Similar to DWRO, the advantage of maintaining the logical schema as ontology is that it can support for future evolution of the DW. The DWO can be formally defined as given below:

DWO = {F,FP,M,D,DIP,RP,L,LP} where,

- F is a set of OWL classes representing the fact;
- FP is a set of data properties representing the fact properties;
- M is a set of data properties representing the measures of the fact;
- D is a set of OWL classes representing the dimensions;
- DIP is a set of data properties representing the dimension properties;
- RP is a set of object properties representing the relationship between facts and dimensions and between dimension and level;
- L is a set of OWL classes representing the levels;
- LP is a set of data properties representing the level properties.

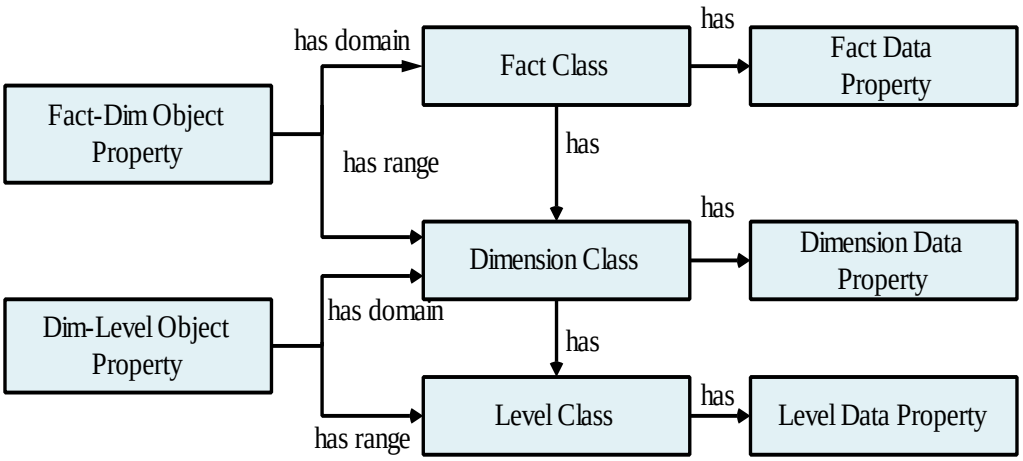

Figure 3.6 Data Warehouse Ontology

3.2.6 Enrichment of the Logical Schema

Before the logical schema is implemented at the physical level, it is further refined in order to satisfy the user needs. For the refinement of the logical schema, the dimensions may be de-normalized and unnecessary attributes can be removed as per the requirements. At this stage, the queries are written based on the requirements. In the refinement process, first the tables involved are extracted from the *from* clause of each query. These tables are then verified with the classes in the

DWO representing the logical schema. The following design rules are applied based on the mapping between the queries and the DWO concepts:

1. When two or more classes in the DWO are mapped to a single table in the query then the classes are merged.
   i.e. **if** $c_i, c_j \rightarrow t_i$ **then** merge (c_i, c_j)
2. When a single class in the DWO is mapped to two or more tables in the query then split operation is performed over the class.
   i.e. **if** $c_i \rightarrow t_i, t_j$ **then** split(c_i)
3. When no mapping exists for a class in the DWO to tables in the query then the corresponding class in DWO is dropped.
   i.e. **if** $\neg c_i \rightarrow \forall(t_i)$ **then** drop(c_i)
4. When no mapping exists for a table in the query to classes in the DWO then the table is added as a new class to the DWO.
   i.e **if** $\neg t_i \rightarrow \forall(c_i)$ **then** add(t_i)

   Where, $c_i, c_j \in C$ i.e classes in DWO and $t_i \in T$ i.e tables in the queries.

A similar operation for refining the attributes needs to be performed. For this the fields contained in the *where* clause of the queries and the table properties in the DWO are mapped as follows:

1. When no mapping exists for a data property in the DWO to the attributes in the query, then the data property is dropped.
   i.e. **if** $\neg dp_i \rightarrow \forall (a_i)$ **then** drop(dp_i)
2. When no mapping exists for an attribute in the query to the data properties in the DWO then the attribute is added to the DWO.
   i.e. **if** $\neg a_i \rightarrow \forall (dp_i)$ **then** add(a_i)

Where, $dp_i \in$ DP i.e data property in DWO and $a_i \in$ A i.e attribute in the queries. After refining DWO using the above operations, the final logical schema for the DW can be generated and represented using graphical notation.

3.2.7 Physical Schema Construction

The final step of OntoMD involves in transformation of logical to physical schema. The physical schema of the DW is defined as follows:

PS = {T,A} Where,

- T is a set of tables representing facts, dimensions and levels;
- A is a set of attributes of the table;
- t_i is a table in T;
- a_i is an attribute in A.

Input : DWO
Output : Relational Data Warehouse

1 **for each** class $c_i \in DWO$ **do**
2 **Create** table t_i from c_i;
3 **Create** the primary key a_i of t_i;
4 **end for**
5 **for each** data property $dp_i \in DP$ **do**
6 $c_j := Dom(dp_i)$;
7 **Find** table t_j;
8 $rng := Rng(dp_i)$;
9 **Create** a_i from dp_i and add to t_j;
10 **end for**
11 **for each** object properties $op_i \in OP$ **do**
12 $c_j := Dom(op_i)$;
13 **Find** table t_i
14 $c_i := Rng\ (op_i)$
15 **Find** table t_i
16 **Add** attribute a_i as foreign key to t_j
17 **Add** foreign key constraint to a_i
which references to primary key of t_i
18 **end for**

Figure 3.7 Physical Schema Construction

From the DWO the physical schema is created in the underlying database using the steps defined in Figure 3.7. The notations used in this algorithm have been explained in section 3.2.1. A table is created for every class in the ontology, with the same name as the OWL class. An attribute representing the primary key is added into this table (Steps 1-4). For all data type properties the domain and the range classes are obtained. An attribute is created with the same name as the data type property. The attribute is added to the table which is mapped to the domain class, and the attribute data type is obtained from the range of the data type property (Steps 5-10).

For an object type property, a foreign key relationship is created between the table that maps the domain class and the table that maps the range class. An attribute is added to the table that maps the domain class and the foreign key constraint is added to this attribute as well (Steps 11-18).

Thus, the OntoMD approach, starting from ontology representation of requirements and data source automatically derives the DW multidimensional schema following conceptual, logical and physical phases of the design. In order to enable the DW designer to construct the schema an OntoMD tool has been developed which is explained in the next section.

3.3 THE ONTOMD TOOL

This section presents the details about the proposed OntoMD tool for DW schema design. The OntoMD tool follows the steps of OntoMD approach systematically and generates the multidimensional schema effectively. This tool assists the designer to reconcile the knowledge in the requirements and the sources at the early stage of design. It also helps to overcome the difficulties in the design process and develop a DW schema in the underlying database following conceptual, logical and physical phases of the DW design. Various components of the OntoMD tool, technology and tools required for implementation of OntoMD tool and comparison with existing tools are discussed in this section.

3.3.1 Components

Figure 3.8 represents the three main components of OntoMD tool which are explained below:

Ontology Representation: The two main inputs of the design process are the requirements and source which are represented in ontology format. The DWRO is constructed from requirement specification using Protégé (Ontology, 2007) which is an ontology editing tool. The local ontologies representing different sources are integrated to DSO using Prompt tool (Noy and Mason, 2003), ontology mapping and merging plug-in for Protégé (Ontology, 2007). Prompt and Protégé are included as plug-ins for the *OntoMD* tool.

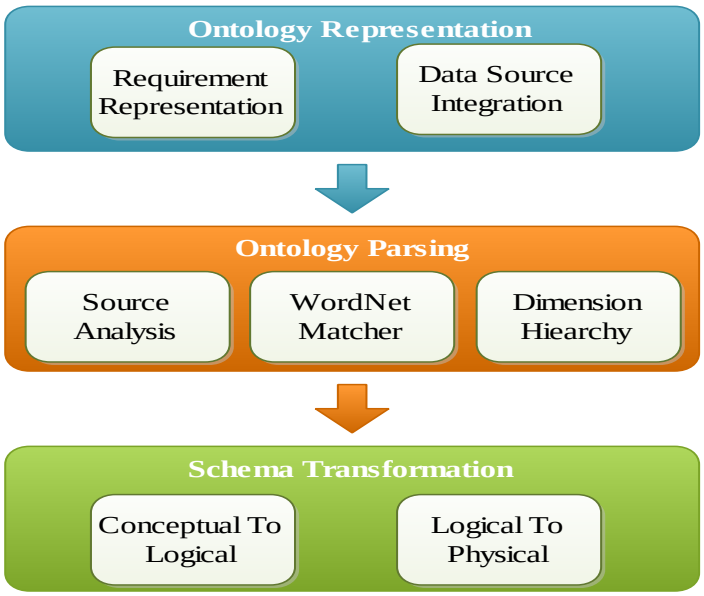

Figure 3.8 Components of OntoMD Tool

Ontology Parsing: For performing *Source analysis*, the DSO is taken as input. Jena API is used for parsing the ontology and to derive the facts, measures, and dimensions based on FactDim algorithm given in section 3.2.2. The results are then displayed to the user. To reconcile the requirements and the source *Wordnet matcher* is used. WordNet (Howe, 2009) matching algorithms take the MDC from

requirements and source. The matched MDC are derived and displayed to the user. The designer can derive the different levels of a dimension based on his choice. To derive the *dimension hierarchy* Jena API is used for parsing the source ontology based on the DimHierarchy algorithm given in section 3.3.4. The conceptual schema elements with dimension hierarchy are displayed in the graphical notation.

Schema Transformation: To derive the logical schema, this component takes the conceptual schema elements as input and represents the fact and dimension in DWO format. The refinement rules explained in section 3.2.6 applied to the logical schema are implemented using Java and Jena API. From the logical schema i.e. DWO, the relational schema is constructed in the underlying database using the physical schema construction steps given in section 3.2.7, which is implemented using Jena API.

3.3.2 Implementation Support

In this section, the general technologies, programming languages and development tools that are used during the implementation process are briefed below:

- **Net Beans 6.9.1 as development environment – IDE**
  The complete development of the tool is realized inside Net Beans 6.9.1. The reason of choosing this development tool can be mainly justified by the fact that it has better support for drag and drop GUI development.

- **Java**
  For implementation of OntoMD tool, Java programming language has been used. Java is a general-purpose, concurrent, class-based, object-oriented computer programming language that is specifically designed to have as few implementation dependencies as possible. The main reason behind this choice is its platform-independent and portable nature.

- **OWL - Web Ontology Language**

  Inside the OntoMD tool, ontologies are chosen for representing the requirements and the data source. As the system needs to read and infer relations from the data source, OWL has been chosen for representing the ontology. The advantage of OWL is that it provides the system with the means of only representing information, but also for automatic processing of that information.

- **Jena API:**

  For reading and parsing the ontology represented in OWL, JENA (Semantic Web Framework for Java) has been chosen for the OntoMD tool. JENA is open source technology and it represents a Java framework for building Semantic Web applications. It provides a Java programmatic environment for both creation and parsing of various Semantic Web Standards (RDF, RDFS, OWL etc.).

- **Protégé**

  Protégé is a free, open-source platform that supports the creation, visualization, and manipulation of ontologies in various representation formats. It is added as a plug-in to OntoMD tool for editing the ontologies.

- **WordNet**

  WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. It is used for finding out the syntactic and semantic relatedness between terms from requirements and data source ontologies.

3.3.3 Comparison of OntoMD with existing tools

A comparison of OntoMD tool with existing tools for DW schema design is given in this section. The various features that are compared are represented in Table 3.1. The main advantage of the proposed OntoMD tool is that it provides a thorough analysis of data source and a formal way of mapping requirements with it to derive the multidimensional model for the DW. The proposed tool provides a good level of automation covering different stages of the design. The relational schema for the DW is automatically constructed from the ontology representation. The user or designer suggestions are utilized while refining the requirements and choosing the dimension hierarchy.

Table 3.1 Comparison of Design Tools

Tool Features	AMDO Tool	GEM Tool	OBDWD Tool	OntoMD tool
Source Analysis	Yes	No	No	Yes
Requirements and Source Mapping	No	Yes	Yes	Yes
Automation of each design Phase	No	No	No	Yes
Relational Schema Construction	No	No	Yes	Yes
User Suggestion	No	Yes	No	Yes

Thus, the proposed tool based on the OntoMD approach improves the design task in a significant way. To illustrate the practical application and to evaluate the proposed OntoMD approach and the corresponding OntoMD tool, the Transaction Processing Council Benchmark H (TPC-H for short) has been chosen (Council, 2008). For the given business domain, the DW schema is generated by applying the various steps of OntoMD approach which has been explained in the following case study.

3.4 CASE STUDY – TPC-H

The TPC-H is a decision support benchmark developed by the TPP Council (Council, 2005). This benchmark was designed to represent a real-world information system and have been widely chosen for its industry-wide relevance. It consists of a set of tables and business oriented ad hoc queries. TPC-H does not represent the activity of any particular business segment, but rather any industry which, sell, or distribute a product worldwide, for example, food distribution, parts, suppliers, etc. (Council, 2005). The purpose of this benchmark is to reduce the diversity of operations found in an information analysis application, while retaining the application's essential performance characteristics.

The TPC-H schema represents ordering and selling activity and it consists of tables such as Lineitem, Orders, Partsupp, Part, Supplier, Customer, etc. Various business analysis that can be performed over the TPC-H domain are profit and revenue analysis, pricing and promotions analysis, Supply and demand analysis etc.

In this section, a detailed discussion is given for each step of OntoMD approach explained in section 3.2, that are applied to the case study. Based on the obtained results the proposed approach is evaluated, which has been provided in the next section. Following are the steps considered for the given TPC-H domain:

1. Representation of Business Requirements and Data Source
2. Analysis of the Data Source
3. Reconciliation of Data Source and Requirements
4. Derivation of Dimension Hierarchy
5. Generation of Logical Schema
6. Enrichment of the Logical Schema

3.4.1 Representation of Business Requirements and Data Source

The requirement specification is derived from the high-level descriptions of the business questions available in TPC-H. A sample of the descriptions is given below:

- Measure the revenue increase by eliminating various ranges of discounts in given product order quantity intervals shipped in a given year;
- Compare revenues for certain product classes and suppliers in a certain region, grouped by more restrictive product classes and all years of orders;
- Retrieve total revenue for lineorder transactions within a given region in a certain time period, grouped by customer nation, supplier nation and year;
- Analyze the profit change in a certain time period, grouped by supplier nation and product category.

For the proposed OntoMD approach it is assumed that the requirement specification is available, which has been already derived using goal-oriented requirement analysis. Figure 3.9 represents the ontology format of the requirement specification for profit and revenue analysis.

The *strategic goal* of the given business domain is to “increase profit for items shipped”. In order to achieve this goal, the *decision goals* that are to be taken are “increase revenue for customer orders” and “increase revenue for parts sold”. In order to satisfy the given *decision goals*, it is necessary to have *information goals* such as “study revenue based on customer and order date” and “study revenue based on parts sold, supplier and date”.

Based on the *information goals*, different analysis needs to be performed, such as “analyze revenue based on customer”, “analyze revenue based on supplier”, and so on. Here, the *context* is “customer” and *measure* is “revenue” for the

information requirement “analyze revenue based on customer”. Similarly, for other *information requirements*, the *contexts* and *measures* can be obtained.

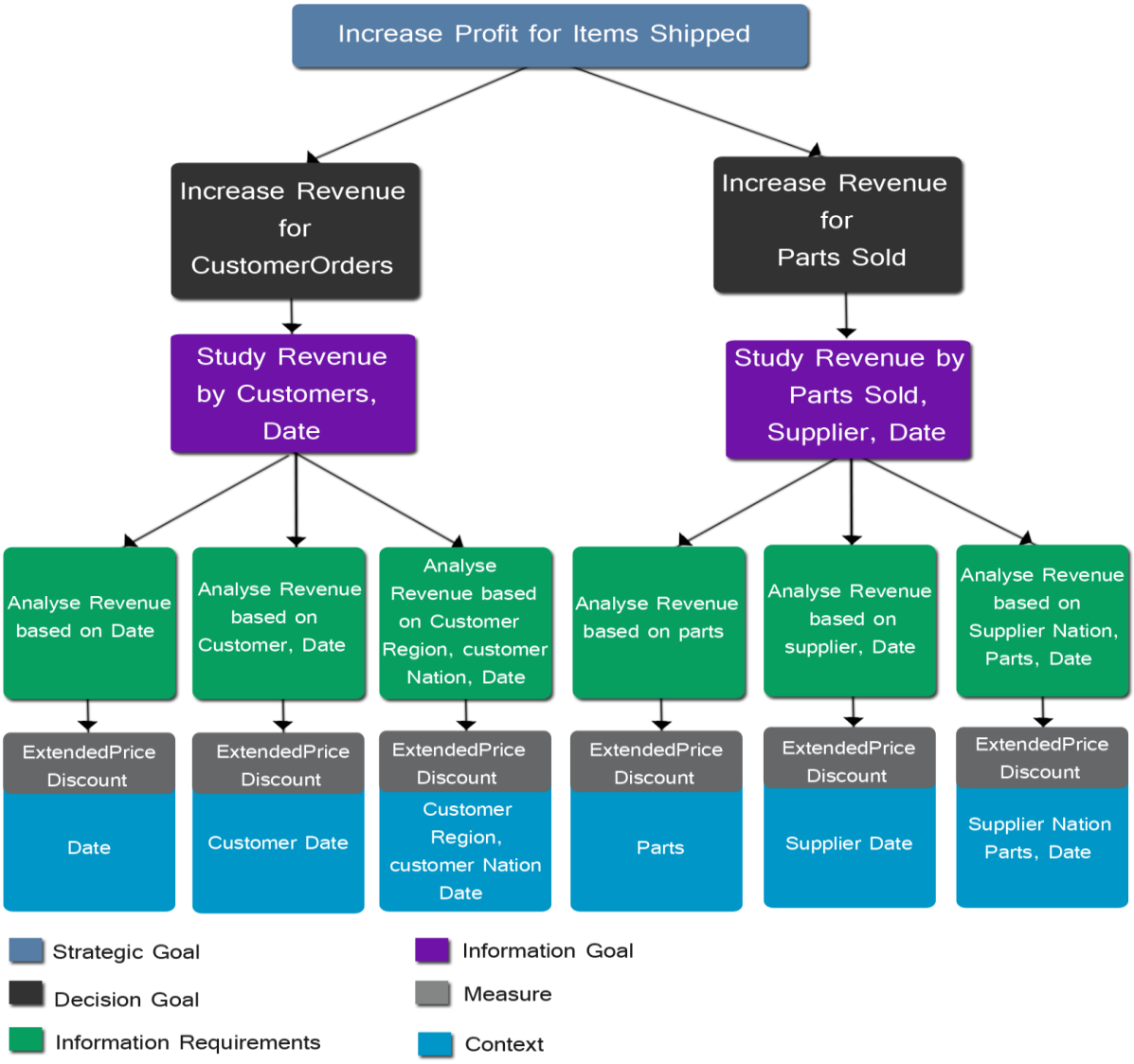

Figure 3.9 Requirement Specification for TPC-H

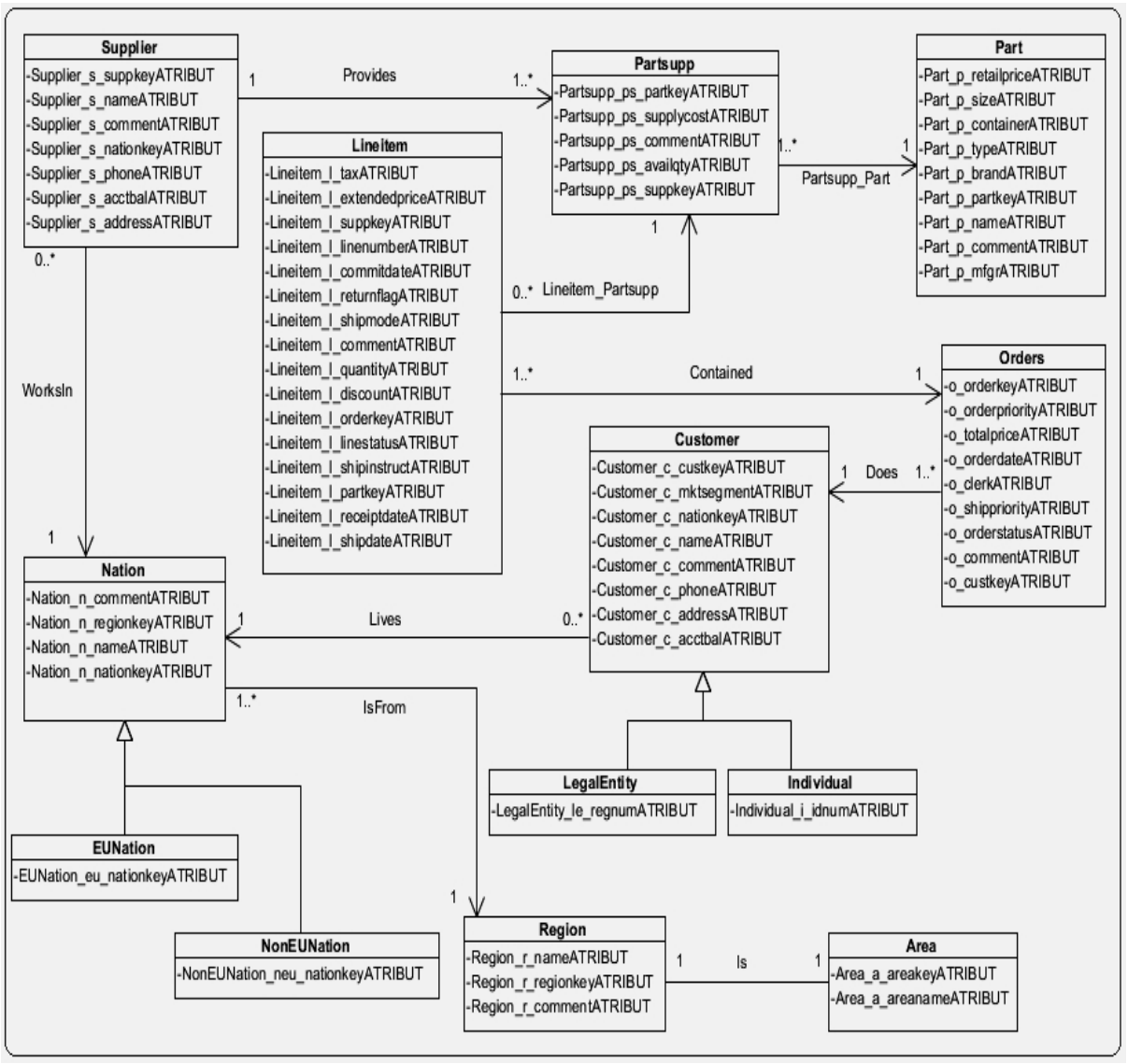

Figure 3.10 Ontology for TPC-H schema

Strategic goals, decision goals, information goals, and *information requirements* along with *contexts* and *measures* are represented as classes of the requirement ontology as explained in section 3.2.1. Once the requirements are specified, the next step is to analyze the data source to derive the MDC. An integrated ontology representing the data sources is available for the TPC-H domain. The graphical representation of TPC-H schema ontology (Skoutas *et al.*, 2009) is shown in Figure 3.10.

3.4.2 Analysis of the Data Source

In order to perform the source analysis, the TPC-H schema ontology is taken as input and the MDC are automatically derived using the FactDim Algorithm given in section 3.2.2. The MDC such as facts, measures, and dimensions obtained after source analysis are given in Table 3.2.

Table 3.2 MD Concepts Derived after Source Analysis

Fact	Measures	Dimensions	Fact	Measures	Dimensions
Orders	Orders_o_totpriceATRIBUT Orders_o_orderkeyATRIBUT Orders_o_orderstatusATRIBUT Orders_o_shippriorityATRIBUT Orders_o_orderpriorityATRIBUT Orders_o_custkeyATRIBUT	-	Individual	Individual_i_idnumATRIBUT	-
EUNation	EUNation_eu_eukeyATRIBUT	-	Parts	Parts_p_sizeATRIBUT Part_p_retailpriceATRIBUT Part_p_partkeyATRIBUT	-
LegalEntity	LegalEntity_le_regnumATRIBUT	-	Partsupp	Partsupp_ps_suppkeyATRIBUT Partsupp_ps_supplycostATRIBUT Partsupp_ps_partkeyATRIBUT Partssupp_ps_availqtyATRIBUT	Parts
LineItem	Lineitem_l_returnflagATRIUT Lineitem_l_quantityATRIBUT Lineitem_l_taxATRIBUT Lineitem_l_linenumberATRIBUT Lineitem_l_extendedpriceATRIBUT Lineitem_l_suppkeyATRIBUT Lineitem_l_partskeyATRIBUT Lineitem_l_discountATRIBUT Lineitem_l_orderkeyATRIBUT	Orders Partsupp	Nation	Nation_n_nationkeyATRIBUT Nation_n_regionkeyATRIBUT	Region
NonEuNation	NonEUNation_neu_neukeyATRIBUT	-	Customer	Customer_c_nationkeyATRIBUT Customer_c_phoneATRIBUT Customer_c_accbalATRIBUT Customer_c_cuskeyATRIBUT	Orders Nation
Supplier	Supplier_s_phoneATRIBUT Supplier_s_accbalATRIBUT Supplier_s_suppkeyATRIBUT Supplier_s_nationkeyATRIBUT	Nation Partsupp	Region	Region_r_regionkeyATRIBUT	Area
			Area	Area_a_areakeyATTRIBUT	-

By setting the threshold value greater than 0 for the ratio *rnp* in FactDim Algorithm, concepts with one or more numerical properties are derived as fact. For the TPC-H schema ontology 13 facts are obtained, such as Orders, LineItem, Supplier, etc. Concepts related to the fact with at-least and at-most one object property is derived as dimensions.

For example, Orders and Partsupp are derived as dimensions for the fact LineItem. After the source is analyzed, the MDC derived are presented to the designer. Here, it is observed that properties such as phone_number, return_flag, etc., are also included as measure concepts as they belong to numerical category. Hence, the results need to be filtered according to the requirements, as all the concepts may not be useful for constructing the DW schema.

3.3.3 Reconciliation of Data Source and Requirements

This step is used to reconcile the results obtained in the previous step with the requirements from requirement ontology. From Table 3.1, the fact LineItem consisting of the measures ExtendedPrice and Discount along with dimensions Orders and Partsupp matches with the measures ExtendedPrice and Discount given in the requirement ontology. The results after matching are shown in Table 3.3. Context such as Customer, Nation, Supplier, Part available in the requirements are not obtained in the filtered result. As these concepts also need to be included in the DW schema, the levels of each dimension are extracted in the next step.

Table 3.3 Results after matching between requirement and source concepts

Fact	Measure	Dimension
LineItem	Lineitem_l_extendedpriceATRIBUT Lineitem_l_discountATRIBUT	Orders Partsupp

3.4.3 Derivation of Dimension Hierarchy

For dimension Orders and Partsupp, the computed hierarchy is shown in Figure 3.11. Customer class having n:1 relationship with Orders dimension becomes the first level. By means of recursively traversing Customer class, Nation is derived as the second level for Orders dimension. In a similar manner, the levels of Partsupp dimension are obtained. At this stage, the designer can choose the levels to be included in the conceptual schema based on the requirements. As the requirement contains Customer, Nation, for Orders dimension and Part, Supplier, Nation and Region for Partsupp dimension, these levels are considered for the DW schema representation. The resulting conceptual schema is shown in Figure 3.10. Here, LineItem is the fact concept, ExtendedPrice and Discount are the measure of the fact, and Orders and Partsupp along with the levels forms the dimension hierarchy.

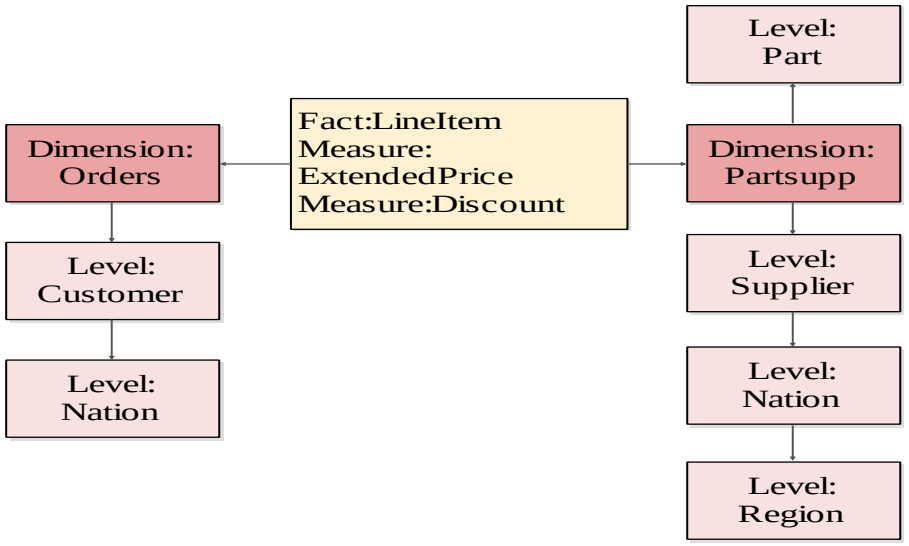

Figure 3.11 Conceptual Schema Representation

3.4.4 Generation of Logical Schema

Following conceptual representation, the logical schema is derived using the steps given in section 3.2.4. Figure 3.12 represents the generated DWO representing the logical schema.

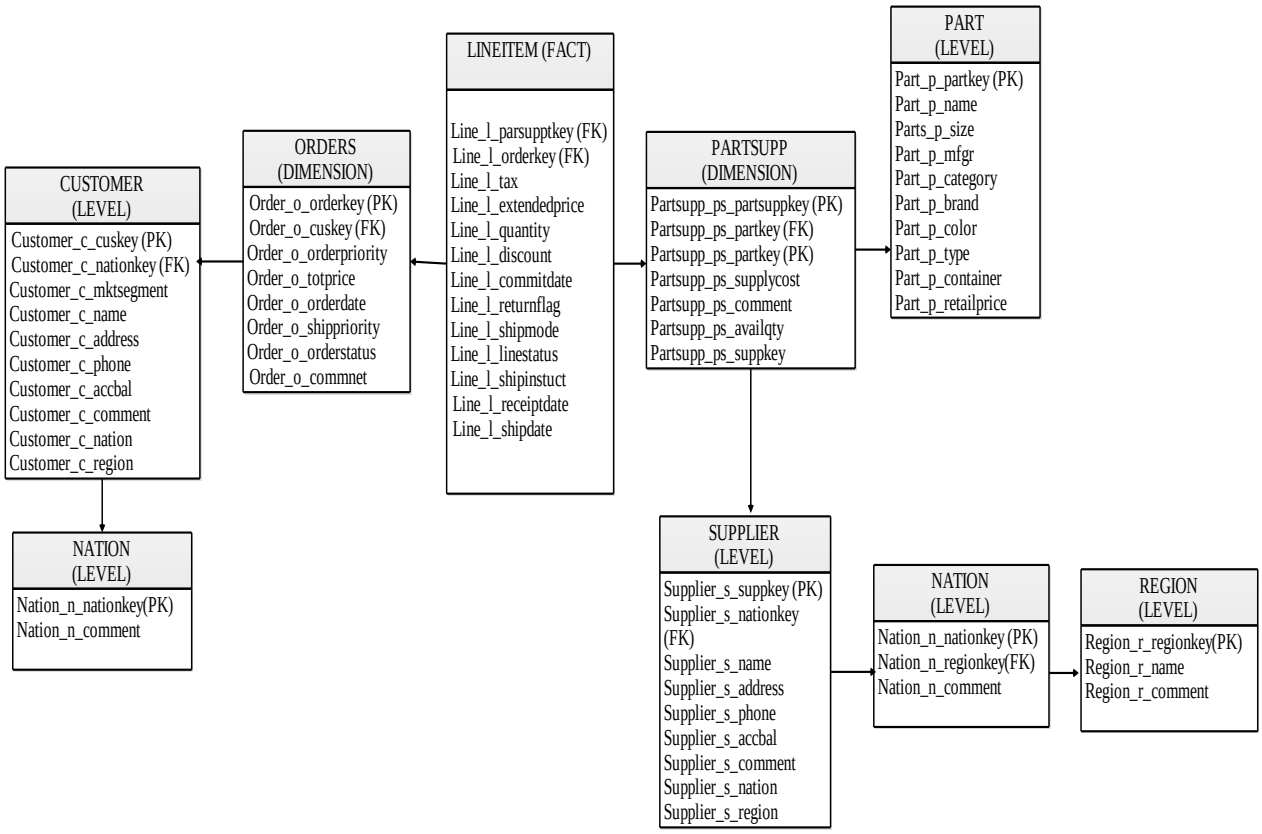

Figure 3.12 Data Warehouse Ontology for Logical Schema

3.4.5 Enrichment of the Logical Schema

In order to refine the logical schema, the queries based on the information requirements are used. For the case study, the Star Schema Benchmark (SSB) queries (O'Neil *et al.,* 2007) are used. SSB is a variation of the TPC-H benchmark, which models the DW for the TPC-H schema. For the TPC-H example, the mapping between logical schema tables to query tables is shown in Figure 3.13. Here, the LineItem and Orders table from logical schema are mapped to Line Order table

contained in the query. Thus, they can be merged to form a single table. The table Partsupp in the logical schema is not available in the query. Hence, Partsupp can be dropped while constructing the final DW schema. The table Date, involved in the query and not available in the logical schema can be kept as a separate dimension table in the final schema. Similar steps can be followed for other mappings. The designer may also need to filter the attributes, as the final schema does not involve all the attributes available. Like table matching, attributes matching can be performed to derive the required result.

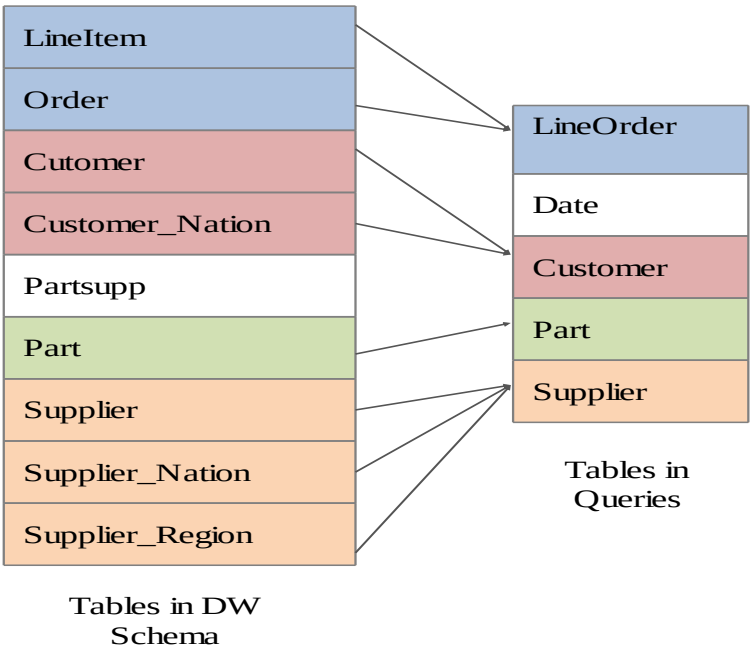

Figure 3.13 Mapping between Tables in Logical Schema and Queries

After the logical schema is refined using the above operations, the final logical schema is generated. Figure 3.14 shows the DWO representing the logical schema produced by the proposed approach. After merging LineItem and Orders table, the Line Order is obtained as the fact table. Similarly, Customer, Part, Supplier, and Date become the dimension table. The primary key and foreign keys are represented for each table of the logical schema. The physical schema can be

automatically constructed from DWO using the steps given in section 3.2.7 and can be evaluated for its quality as discussed in the following section.

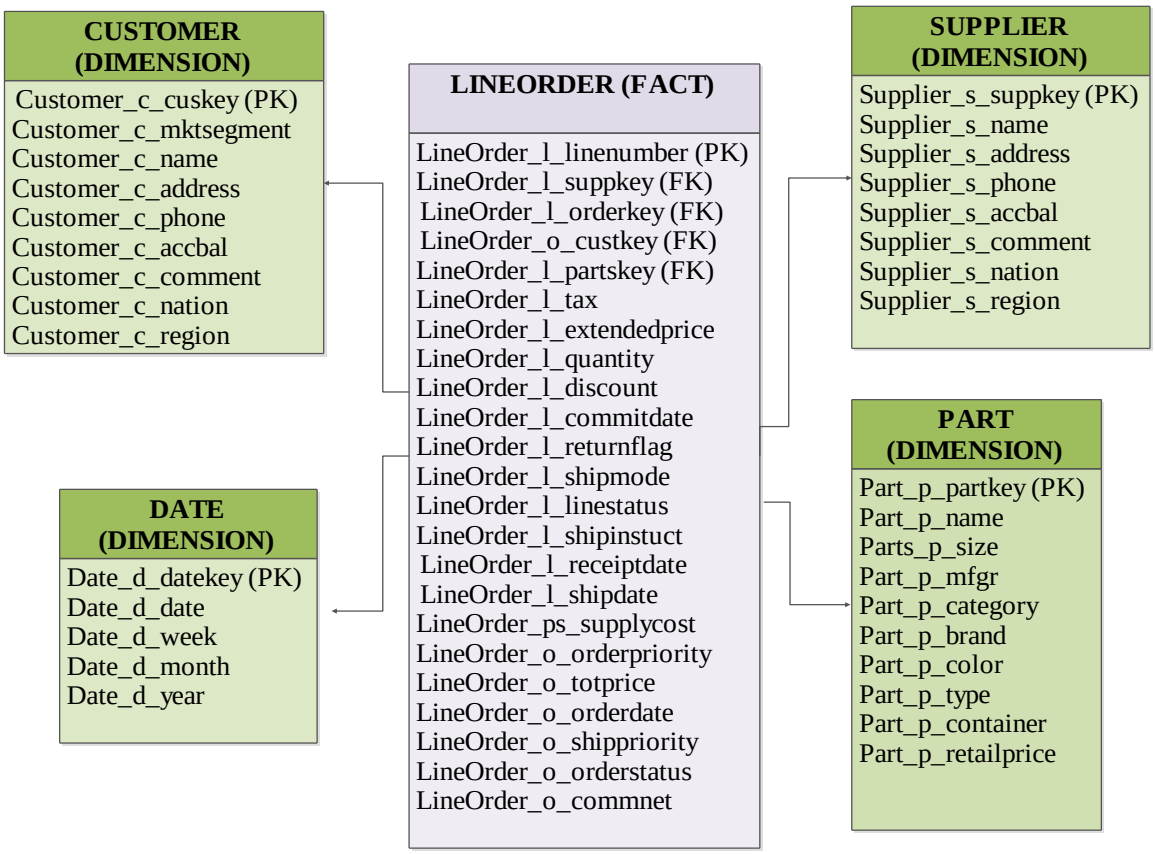

Figure 3.14 Logical Schema for Data Warehouse

3.5 RESULTS AND DISCUSSION

This section discusses about the experimental setup for evaluating the effectiveness of the proposed approach along with the schema quality parameters used for the evaluation. Further, the results obtained with the proposed OntoMD proposed is compared with other approaches such as AMDO (Romero and Abelló, 2010), GEM (Romero *et al.*, 2011) and OBDW (Selma *et al.*, 2012).

3.5.1 Experimental Setup

To evaluate the proposed approach in terms schema quality three different domains, TPC-H, EU-Car Rental (Frias *et al.,* 2003), and LUMB (Guo *et al.,* 2005) are used. TPC-H benchmark has been explained in section 3.3. EU-Car Rental provides an integrated data source schema, and a set of business requirements for the car rental domain. The supporting schema contains information, such as, rental_agreement, customer, branch, reservation etc. The LUBM is composed of several data sources and a global schema available in an ontology format on a university's domain with concepts such as Undergraduate Student, Assistant Professor etc. and 14 test queries. The DW schemas for these three domains are generated using the OntoMD tool.

In order to validate the schema generated, the physical schema is constructed in the Oracle11G DBMS. In order to execute the queries the DW is populated using the data generator available for each domain.

3.5.2 Schema Quality Metrics

The effectiveness of the proposed approach is evaluated in terms of the quality of the output schema generated. Following are the parameters used to verify the DW schema quality (Vassiliadis, 2000):

1. **Correctness**: Final inspection of DW schema for each entity and its corresponding ones in the sources.

$$\text{Correctness} = \frac{(TE - NUM)}{TE} \times 100 \quad \text{..... (3.1)}$$

Where, TE is the total number of entities and NUM is the number of unmatched entities.

2. **Completeness**: Final inspection of DW schema for useful entities in the sources, not represented in the DW schema.

$$\text{Completeness} = \frac{(TE - NNU)}{TE} \times 100 \quad \text{..... (3.2)}$$

Where, TE is the total number of entities and NNU is the number of non-useful entities.

3. **Minimality**: Final inspection of DW schema for undesired redundant information.

$$\text{Minimality} = \frac{(TE - NR)}{TE} \times 100 \quad \text{..... (3.3)}$$

Where, TE is the total number of entities and NR is the number of redundant entities.

4. **Traceability**: Inspection of DW schema for inability to cover user requirements.

$$\text{Traceability} = \frac{(TE - NNR)}{TE} \times 100 \quad \text{..... (3.4)}$$

Where, TE is the total number of entities and NNR is the number of entities not covered the requirements.

5. **Interpretability**: Mapping of conceptual to logical entities and from logical to physical entities.

$$\text{Interpretability} = \frac{(TE - NNT)}{TE} \times 100 \quad \text{..... (3.5)}$$

Where, TE is the total number of entities and NNT is the number of non-traceable entities.

3.5.3 Result Analysis

The values of the given schema quality parameters are computed for the generated DW schema for the three given domains. The results are compared for the proposed OntoMD approach with the existing approaches. Table 3.4 provides the obtained results for each quality metric.

Table 3.4 Results for schema quality generated by ontology based approaches

Schema Quality Metrics	Domains	AMDO	GEM	OBDW	OntoMD
Correctness %	TPC-H	85	89	90	100
	CAR-RENTAL	86	87	89	100
	LUMB	84	85	90	100
Completeness %	TPC-H	78	83	85	98
	CAR-RENTAL	75	82	87	98
	LUMB	75	83	85	97
Minimality %	TPC-H	82	85	90	100
	CAR-RENTAL	80	86	89	100
	LUMB	80	85	89	100
Traceability %	TPC-H	75	86	100	100
	CAR-RENTAL	73	84	100	100
	LUMB	75	86	100	100
Interpretability %	TPC-H	72	80	87	95
	CAR-RENTAL	70	82	88	96
	LUMB	70	80	88	95

Figure 3.15 presents the comparison of correctness metrics of ontology-based approaches for the three domains. A formal reconciliation of requirements and data source has been carried out by the proposed OntoMD approach. Hence, the correctness achieved is 100% as the DW schema entities produced have a mapping with the corresponding source schema entities for all three domains.

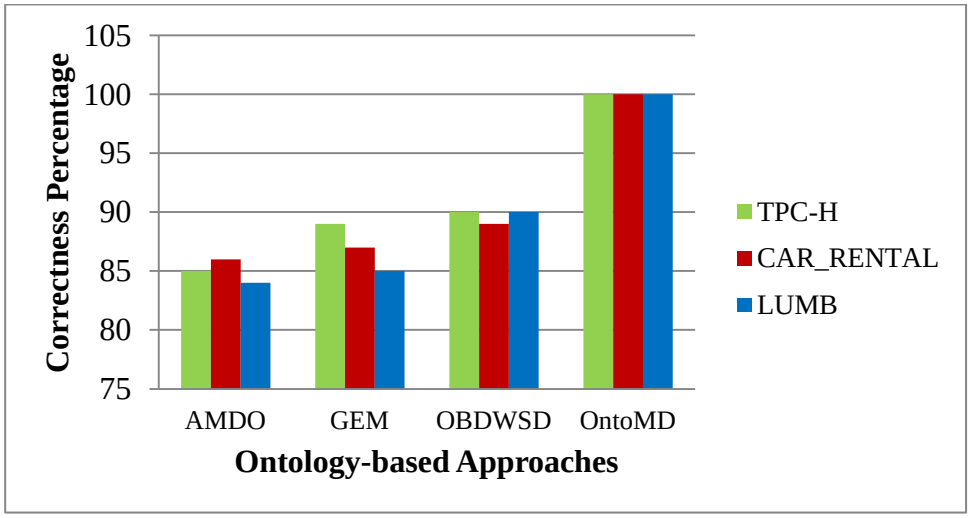

Figure 3.15 Correctness Analysis

The useful entity in the source schema not present in the DW schema for TPC-H domain is *customer type* in customer dimension, hence the achieved completeness is 98%. From the Figure 3.16 it was observed that the results for completeness metrics for other domains are also better when compared to the existing approaches.

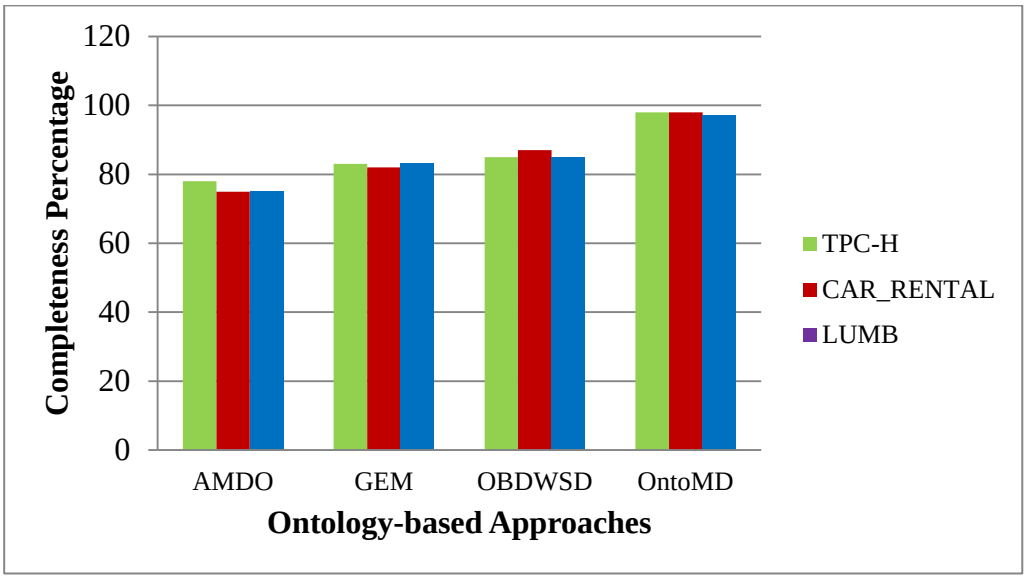

Figure 3.16 Completeness Analysis

Figure 3.17 analyses the effectiveness of the approaches in terms of minimizing the redundant entities in the output schema. The proposed approach performs filtering of the required MDC from the source. Hence, the DW schema produced by OntoMD does not contain any redundant entities and thus the minimality achieved is 100% for the given domains.

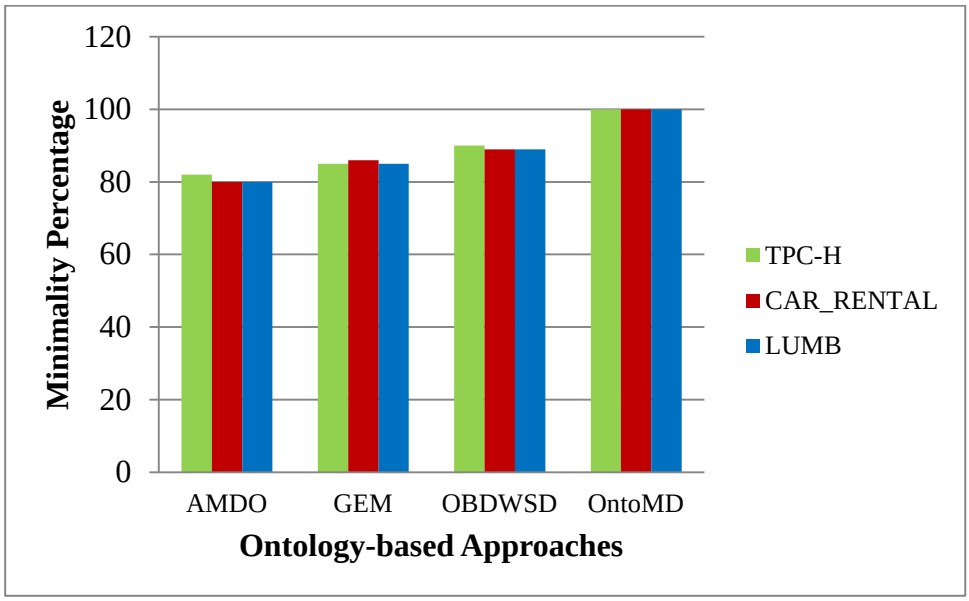

Figure 3.17 Minimality Analysis

From the Figure 3.18 it has been observed that the OBDW and the proposed OntoMD achieved traceability as 100% for the three domains. The reason is that these two approaches successfully execute the given set of queries, which in turn satisfy the given set of requirements.

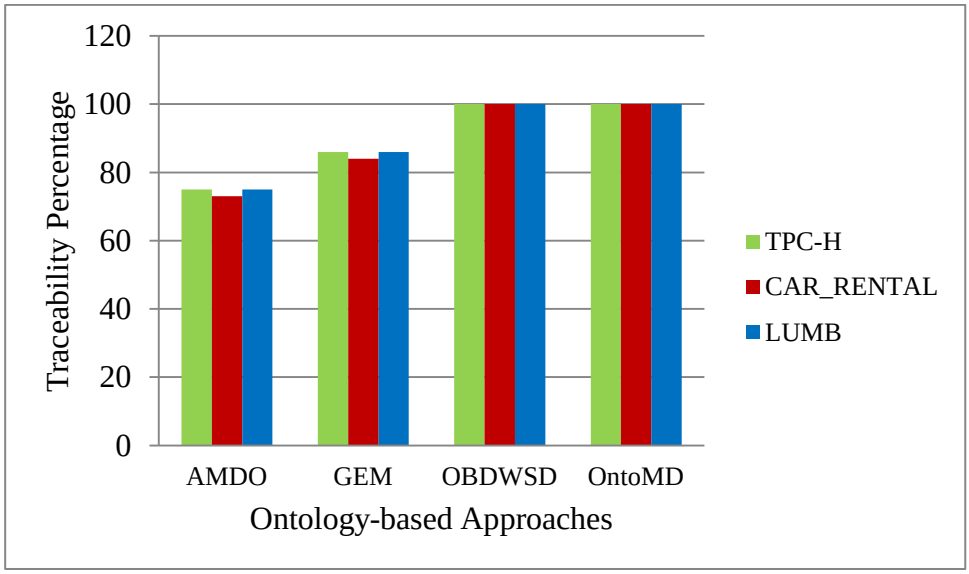

Figure 3.18 Traceability Analysis

In the proposed OntoMD design approach, most of the entities from conceptual to logical and, logical to physical schema could be traced and hence it could achieve 92% as interpretability for TPC-H domain. This result is comparatively high as shown in Figure 3.19, when compared to existing approaches for the given domains.

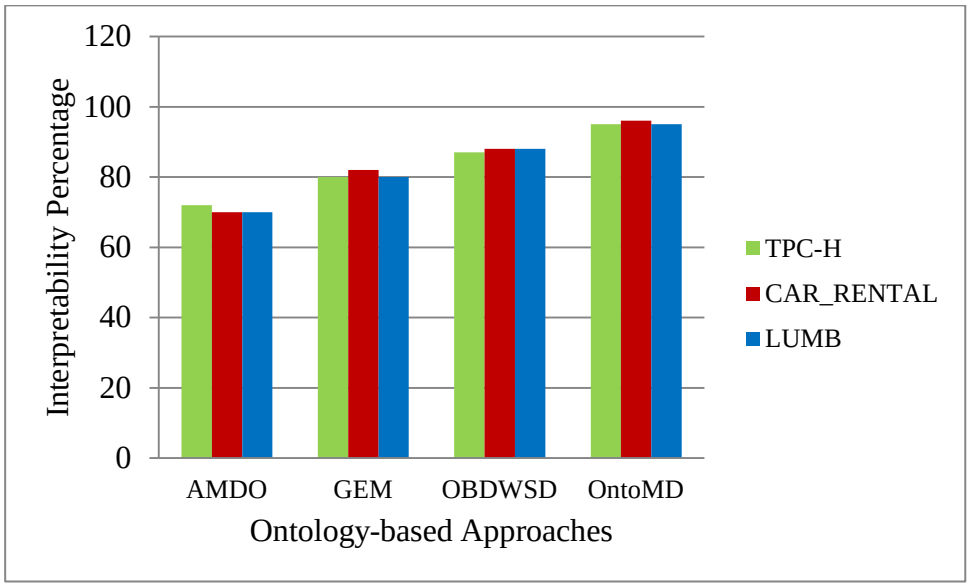

Figure 3.19 Interpretability Analysis

From the above results it has been observed that the OBDW approach and proposed work provide good results compared to other approaches. The proposed work outperforms OBDW and other approaches w.r.t correctness, completeness and minimality, and interpretability. The following inference is made from the Figure 3.15 to Figure 3.19:

i. The proposed OntoMD achieve 10% more result for correctness when compared to OBDW, 13% more than GEM and 15% more result than AMDO.

ii. For completeness OntoMD produce 13% better results compared to OBDW, 16% more than GEM and 22% more result than AMDO.

iii. In case of minimality OntoMD achieve 10% more result when compared to OBDW, 53% more than GEM and 19% more result than AMDO.

iv. OntoMD and OBDW achieve same results for traceability. Whereas, OntoMD produce 14% more result when compared to GEM and 25% more than AMDO.

v. For interpretability OntoMD produce 13% better results compared to OBDW, 16% more than GEM and 22% more result than AMDO.

3.6 SUMMARY

In this chapter the proposed ontology-based approach for the DW schema design has been described. In this approach, utilizing semantic web tools such as ontology, a conceptual design phase has been used, which aimed at reconciliation of MDC present in requirements and data source in a formal way. The resulting conceptual schema was then mapped into a logical multidimensional schema, followed by an enrichment of this schema using SQL queries. Based on the proposed implementation, a tool called OntoMD was developed to facilitate the designer for the design task. The design method was applied to a case study. In order to validate the proposal, the physical schema was implemented using TPC-H benchmark and other domains. By comparing the proposed work with existing approaches, it was inferred that the OntoMD produced a good level of automation of the design task. It helps to reduce the burden of designer to perform manual reconciliation and redesign involved during the design process. Moreover, the quality of the output generated by the proposed approach outperforms the existing ontological approaches. In the next chapter, the focus is on the impact of changes in data source and business requirements of the DW schema and its dependent entities.

CHAPTER 4

ONTOLOGICAL APPROACH TO HANDLE DATA WAREHOUSE SCHEMA EVOLUTION

4.1 INTRODUCTION

DWs tend to evolve, due to changes in data sources and business requirements of users. Both these changes result in DW schema evolution. These changes are handled either by just updating it in the DW structure (Benitez-Guerrero *et al.,* 2004; Curino *et al.,* 2009), or it is developed as a new version (Bebel *et al.,* 2006; Sahpaski *et al.,* 2009).

Existing approaches in DW schema evolution concentrate on source changes and few over requirements changes. As these changes are updated at the physical level of the DW schema, it may induce high maintenance costs and complex OLAP server administration. Moreover, any change in the DW schema structure may have an impact on its dependent entities. Existing research over DW evolution do not focus on the impact analysis. In this research an ontological approach to automate the evolution (OntoEvol) of the DW schema has been proposed. This method assists the DW designer to handle evolution and also analyze the change impact, based on which decision can be made to carry out the changes at the physical level.

Section 4.2 of this chapter provides the details about the proposed approach along with the steps involved. The automatic adaptation part of the proposed system has been detailed in section 4.3. Application of the OntoEvol approach to a case study has been explained in section 4.4. Section 4.5 provides the evaluation of the proposed approach and comparison with existing approaches. Summary of the chapter is provided in section 4.6.

4.2 OntoEvol: PROPOSED ONTOLOGICAL EVOLUTION APPROACH

This section describes the formal approach that has been developed for managing the DW schema evolution using ontology. When data source and requirements evolves the DW schema need to be updated in order to provide up-to-date information to users. Before making structural changes to the existing DW physical schema the proposed work provides a method for updating the conceptual representation of the schema. Hence, the data source schema, requirements and DW schema are represented in ontology format. Given information about the changes in the data source or requirements, the proposed OntoEvol approach produces the updated version of the DW schema at the conceptual level, and based on the impact of change the DW physical schema is constructed.

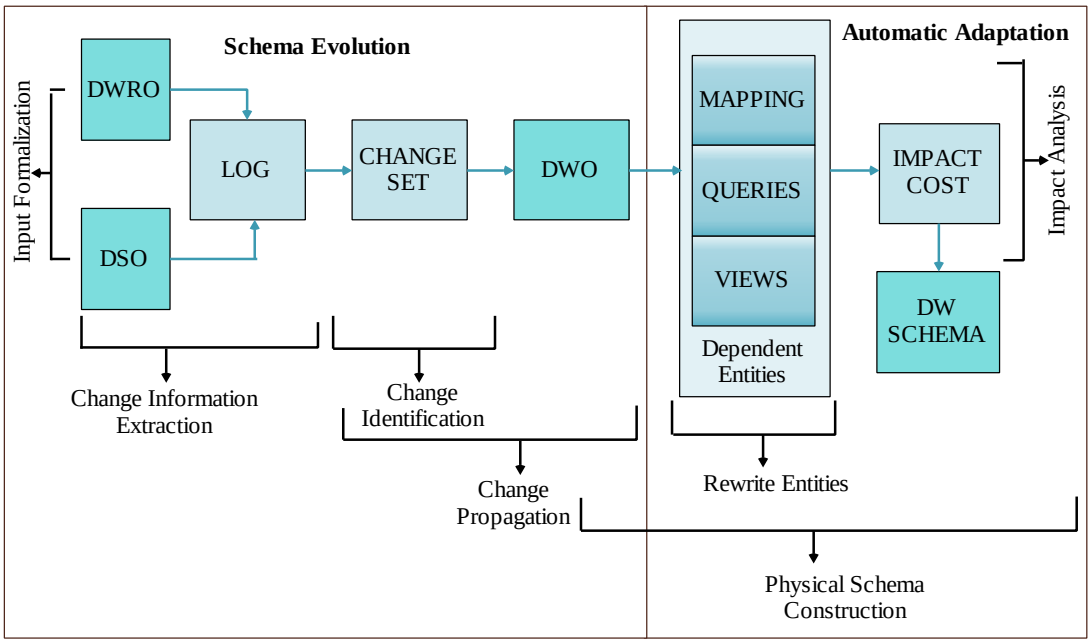

Figure 4.1 OntoEvol System

Figure 4.1 represents the steps followed in the proposed approach. By making use of the ontological representation of the inputs, automation (semi- automation) of the evolution task has been achieved. In the *schema evolution* part of the OntoEvol approach, the changes that occurred at the data source or requirements are extracted

from the corresponding ontology representation. The type of change and the element affected by the change are derived and the change is propagated to the DW schema. As the DW schema changes, it affects the dependent entities such as the *mapping* between data source and DW schema, *queries* and *views*.

In the *automatic adaptation* part of the proposed system the impact of a given change on the dependent entities is analyzed. The impact of a change is obtained by calculating the cost of mapping adjustment, query rewriting and view rewriting. Based on the total cost obtained the DW administrator can make a decision to perform the changes at the physical level. Finally, from the DW ontology the new version of the DW physical schema is constructed automatically in the underlying database. The *automatic adaptation* part of the proposed system has been discussed in section 4.3. Following are the steps of the *schema evolution* part of the proposed approach which has been detailed in this section:

1. Input Formalization
2. Definition of Evolution Operators
3. Change Information Extraction
4. Change Identification
5. Change Propagation

4.2.1 Input Formalization

This step presents the formalization of inputs of the proposed method in order to standardize and to ensure the correctness of the DW evolution process. OWL ontology is used to describe the semantics of different entities involved in the methodology. The reason for using OWL as the utility, instead of XML, UML or others, is that OWL supports automatic processing of information represented in the ontology. Thus, the data source, requirements and the DW schema are described in the OWL ontology format. As explained in section 3.2.1 of Chapter 3 these entities are formally represented using ontology concepts as given below:

The data source ontology (DSO) is the collection of classes, data type properties, and object properties which is defined as follows:
DSO = {C, DP, OP} where,

- C is a set of OWL classes;
- DP is a set of data properties;
- OP is a set of object properties.

The DW requirement ontology (DWRO) can be defined as:
DWRO = {S, I, D, IR, BP, M, CN} where,

- S is a set of OWL classes representing the strategic goals;
- I is a set of OWL classes representing the information goals;
- D is a set of OWL classes representing the decision goals;
- IR is a set of OWL classes representing the information requirements;
- BP is a set of OWL classes representing business process;
- ME is a set of OWL classes representing measures;
- CN is a set of OWL classes representing the contexts.

The graphical representation of the DWRO is given in Figure 4.2.

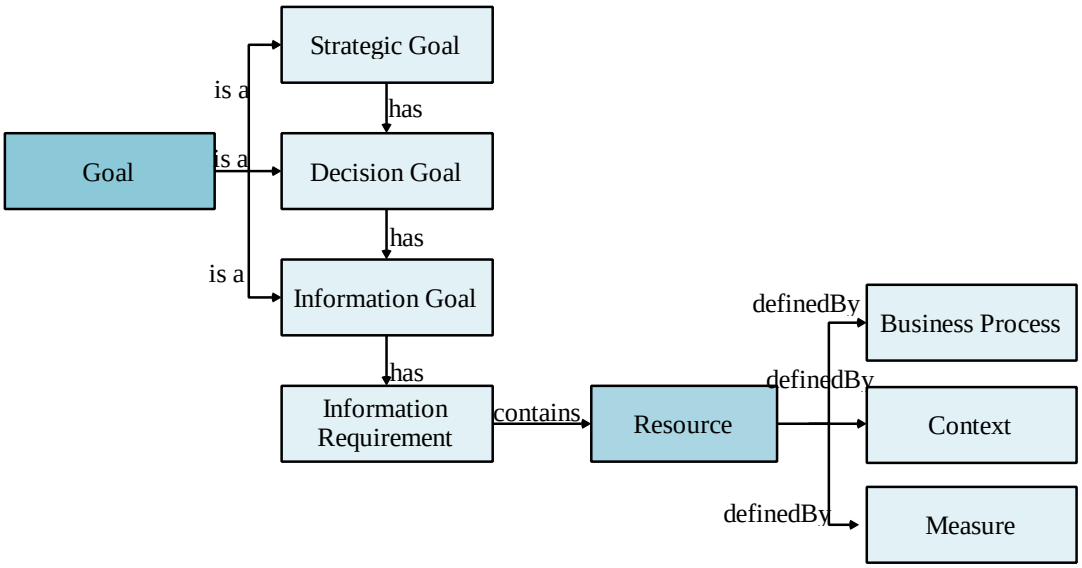

Figure 4.2 Data Warehouse Requirement Ontology

The formal and graphical representation of DW ontology (DWO) explained in section 3.2.5 of chapter 3 used to represent the DW schema has been reproduced below. The DWO can be formally defined as given below:

DWO = {F, FP, M, D, DIP, RP, L, LP}, where,

- F is a set of OWL classes representing the fact;
- FP is a set of data properties representing the fact properties;
- M is a set of data properties representing the measures of the fact;
- D is a set of OWL classes representing the dimensions;
- DIP is a set of data properties representing the dimension properties;
- RP is a set of object properties representing the relationship between facts and dimensions and between dimension and level;
- L is a set of OWL classes representing the levels;
- LP is a set of data properties representing the level properties.

The graphical representation of the DWO is given in Figure 4.3.

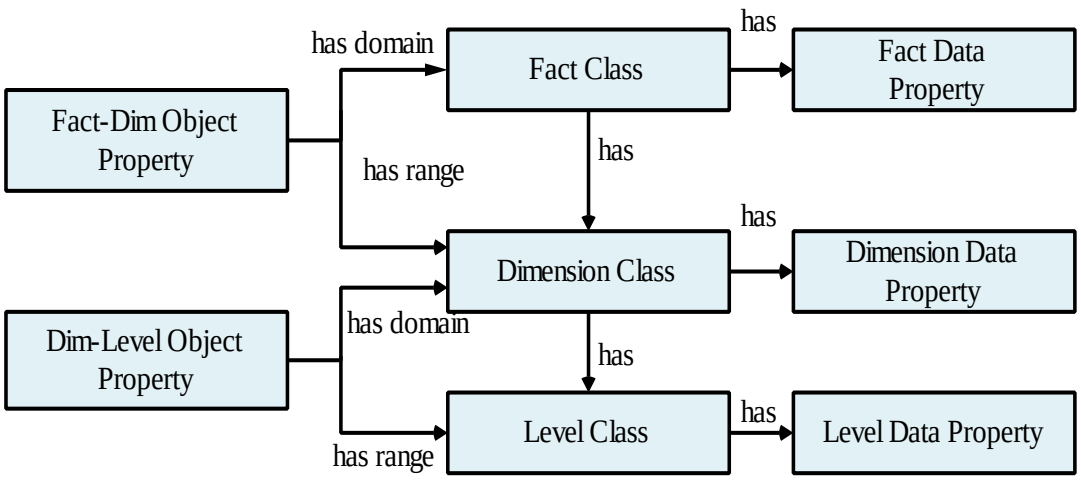

Figure 4.3 Data Warehouse Ontology

4.2.2 Definition of Evolution Operators

The proposed approach defines a set of evolution operators to represent the type of change that occurs over the DW schema. The three possible type of changes that

occur over the DW schema are: *addition*, *deletion* and *rename*. The multidimensional elements of DW such as Fact, Dimension, Measures, Levels, Fact Properties, Dimension Properties etc., are subject to change. As the proposed system uses DWO to represent a DW schema when new changes are carried over the ontology, it requires additional changes to be executed. For example, *addition* of a new dimension i.e., dimension class to the DWO requires *addition* of its data property and object property. The type of change, element changed and additional changes are given in Table 4.1.

Table 4.1 Evolution Operators

Type of Change	**DW Schema Elements**	**Equivalent Ontology Concept Changed**	**Elementary Changes**
Addition	Table (Fact, Dimension)	Class	Add Data Property Add Object Property
	Attribute (Measure, Descriptive)	Data Property	Add Property Domain Add Property Range
	Relationship (Primary Key, Foreign Key)	Object Property	Add Property Domain Add Property Range
Deletion	Table (Fact, Dimension)	Class	Delete Data Property Delete Object Property
	Attribute (Measure, Descriptive)	Data Property	Delete Property Domain Delete Property Range
	Relationship (Primary Key, Foreign Key)	Object Property	Delete Property Domain Delete Property Range
Rename	Table (Fact, Dimension)	Class	Rename Class (If required)
	Attribute (Measure, Descriptive)	Data Property	Rename Data Property (If required)
	Relationship (Primary Key, Foreign Key)	Object Property	Rename Object Property (If required)

4.2.3 Change Information Extraction

The data source schema may be modified, for example, due to the addition of a relation or deletion of an attribute. The changes in the original data source schema are carried over the DSO to enable the evolution task. This is done by means of ontology editing tool such as protégé (Ontology, 2007). Similarly, the requirements of the DW may change, for example, the addition of a new business analysis perspective i.e. a new dimension. As the proposed approach maintains the requirements in ontology format, the changes can be easily incorporated over the DWRO using protégé (Ontology, 2007).

The proposed system begins by extracting any change that has recently occurred over either DSO or the DWRO. The OntoEvol system uses Change Annotation Ontology (ChAO) (Ontology, 2007) which acts as a log to capture the changes that has happened to the ontology. The concept changed (name of the concept), concept type (multidimensional element type) and change type (addition, deletion or rename) are retrieved from ChAO.

4.2.4 Change Identification

This step involves the identification of the multidimensional type of an element when the change type is *addition*. When a new concept is added to the DSO or DWRO, before it is propagated over the DWO, its multidimensional type has to be identified. For example, when a new class is added to the DSO, it may act as a fact, dimension or level in the DW schema. According to the designer's choice before the new class is added to the DWO, its multidimensional type is identified using the proposed algorithm *FindClassType* given in Figure 4.4. The notations used in this algorithm are given in section 4.2.1.

```
FindClassType (DSO, ConceptChanged, MDList)
1   for all ci ∈ C do
2       if ci == ConceptChanged then
3           if (Rng(ci.opi)!= null && Rng(ci.opi) ∈ DimensionList))then
4               for all data properties ci .dpi ∈ DP do
5                   rng := Rng(ci .dpi), rng ∈ DTxml
6                   if isnumeric(rng) then
7                       tn++;
8                   end if
9               end for
10              if tn>threshold then
11                  cj:= Rng(ci.opi), ci ∈ C
12                  if (ci.opi allValueFrom cj && maxCardinality = = 1) then
13                      ci =FactClass
14                  end if
15              endif
16          else if (Domain(ci.opi)! =null && Domain(ci.opi) ∈ FactList) then
17              cj= Domain(ci.opi)
18              if (cj.opi allValueFrom ci && maxCardinality = = 1) then
19                  ci =DimensionClass
20              end if
21          else if (Domain(ci.opi)! =null && Domain(ci.opi) ∈ (DimList ||
                                                          LevelList)) then
22              ci = LevelClass
23          endif
24          end if
25          end if
26      end if
27  end for
```

Figure 4.4 Algorithm FindClassType

The algorithm takes DSO, the new concept changed and the multidimensional lists (MDList) containing *FactList*, *DimList* and *LevelList* of DWO as inputs. First, the algorithm finds whether the class c_i is a fact, for this the range class of c_i is obtained. If the range class exists in DSO and it belongs to *DimList*, c_i is likely to be a fact. The data properties of c_i are derived and checked whether it contains enough numerical properties to qualify as a fact. If c_i has n:1 relationship with range class, then it is identified as fact (steps 2-15). To find whether c_i is a dimension, the

domain class of c_i is obtained. If the domain class c_j belongs to *FactList* then c_i is likely to be a dimension. If the domain class c_i has 1:n relationship with fact then c_i is identified as a dimension (steps 16-20). To find whether c_i is a level, the domain class of c_i is obtained. If the domain class c_j belongs to *DimList* or *LevelList* then c_i is identified as a level (steps 17-23).

The next algorithm *FindDPType* given in Figure 4.5 checks whether the concept added is a data property and identifies it as a fact property, dimension property or level property. It takes DSO, the new property added and the MDList such as *FactList*, *DimList* and *LevelList* of DWO as inputs. From the DSO for the newly added data property dp_i, its domain *d* is obtained (Step 2-3). If *d* is in *FactList* then the concept added is identified as fact property (Steps 4-5). If *d* is in *DimList* then it is identified as dimension property (Steps 6-7). If *d* belongs to *LevelList* then the concept is identified as level property (Steps 8-9).

```
FindDPType (DSO,ConceptChanged, MDList)
1     for all dpi ∈ DP do
2           if dpi == ConceptChanged then
3                 d= Domain(dpi)
4                 if d ∈ FactList then
5                       dpi=FactProperty
6                 else if d= ∈ DimList then
7                       dpi=DimensionProperty
8                 else
9                       dpi=LevelProperty
10                end if
11                end if
12                end if
13          end if
14    end for
```

Figure 4.5 Algorithm FindDPType

Finally, to identify an object property that is added, the proposed algorithm *FindOPType* is used. This algorithm checks whether the concept added is an object

property and identifies it as a fact relation, dimension relation or a level relation. The domain *d* and range *r* of the newly added object property op_i is obtained. If *d* is a fact belonging to *FactList* and *r* is a dimension belonging to *DimList* then concept added is identified as fact-dimension relation. If *d* is a dimension and *r* is a fact then the concept added is identified as dimension-fact relation. If *d* is a dimension and *r* is a level, then the concept added is identified as dimension-level relation. The steps of *FindOPType* are given in Figure 4.6.

```
FindOPType (DSO,ConceptChanged, MDList)
1     for all opi ∈ OP do
2           if opi == ConceptChanged then
3                 d=Domain(opi)
4                 r= Rng(opi)
5                 if (d ∈(FactList) && r ∈ ( DimensionList) then
6                       opi=FactDiemensionRelation
7                 else if (d ∈ ( DimensionList) && r ∈ (FactList) then
8                       opi=DiemensionFactRelation
9                 else if (d ∈ ( DimensionList) && r ∈ (LevelList) then
10                      opi=DiemensionLevelRelation
11                end if
12                end if
13                end if
14          end if
15    end for
```

Figure 4.6 Algorithm FindOPType

4.2.5 Change Propagation

Once the changes are extracted and the type of change is identified, the next step involves in propagating the changes to the DW schema. Depending on the type of change three algorithms are proposed to apply changes over the DWO. In these algorithms, various notations are used which has been explained in section 4.2.1.

If the type of change is addition, the multidimensional element is identified using the previous step. For propagating the addition change, the proposed algorithm *ApplyChangeAddition* given in Figure 4.7 is used. In this algorithm, if the concept type is a class, then the list of data properties and object properties are retrieved for the class. The new class is added to the DWO and for each data property its range and domain are included. Similarly, for each object property its range and domain are included (Steps 1-6). If the concept type is a data property, the new data property is added to the corresponding class in DWO and its range and domain are included (Steps 7-10). If the concept type is an object property, the new object property is added to the class in DWO and its range and domain are included accordingly (Steps 11-14).

```
ApplyChangeAddition(DWO, Concept, ConceptType)
1     if Concept _Type ==Class then
2           c=Concept
3           Add c ∈ C in O
4           Get DP and OP
5           AddDP(c,DP)
6           AddOP(c,OP)
7     else if Concept _Type ==DataProperty then
8           dp= Concept
9           Get  c of dp
10          AddDP(c,dp)
11    else if Concept _Type ==ObjectProperty then
12          op= Concept
13          Get  c of op
14          AddDP(c,OP)
15    end if
16    end if
17    end if
18    AddDP(c,dp)
19          Add ci .dp ∈ DP
20          Set Rng(c.dp)
21          Set Dom(c.dp)
22    end
23    AddOP(c,op)
24          Add c.op ∈ OP
25          Set Rng(c.op)
26          Set Dom(c.op)
27    end
```

Figure 4.7 Algorithm ApplyChangeAddition

For propagating the *deletion* change, the proposed *ApplyChangeDeletion* algorithm given in Figure 4.8 is used. If the concept type is a class, then the class is deleted from DWO. The corresponding data properties and object properties of the class are also deleted (Steps 1-10). If the concept type is data property or object property, it can be directly deleted from the given class (Steps 11-20).

```
ApplyChangeDeletion(DWO, Concept, Concept _Type)
1     if Concept _Type ==Class then
2           c=Concept
3           Delete c ∈ C in O
4           for all data properties c.dp_i ∈ DP do
5                 Delete  c.dp_i
6           end for
7           for all object properties c_i.op_i ∈OP do
8                 Delete  c.op_i
9           end for
10    end if
11    if Concept _Type == DataProperty then
12          dp= Concept
13          Get  c of dp
14          Delete  c.dp
15    end if
16    if Concept _Type == ObjectProperty then
17          op= Concept
18          Get  c of op
19          Delete  c.op
20    end if
```

Figure 4.8 Algorithm ApplyChangeDeletion

For propagating the *rename* change, the proposed algorithm *ApplyChangeRename* given in Figure 4.9 is used. If the concept type is a class, the old concept is deleted and the new concept name is included in the ontology (Steps 1-7). Similar step is followed if the concept type is a data property or an object property (Steps 8-21).

```
ApplyChangeRename(DWO, OldConcept,NewConcept, Concept _Type)
1     if Concept _Type ==Class then
2           for all ci ∈ C do
3                 if ci == OldConcept then
4                       Add NewConcept
5                       Delete ci
6                 end if
7           end for
8     else if Concept _Type ==DataProperty then
9           for all dpi ∈ DP do
10                if dpi == OldConcept then
11                      Add NewConcept
12                      Delete dpi
13                end if
14          end for
15    else if Concept _Type ==ObjectProperty then
16          for all opi ∈ OP do
17                if opi == OldConcept then
18                      Add NewConcept
19                      Delete opi
20                end if
21          end for
22    end if
23    end if
24    end if
```

Figure 4.9 Algorithm ApplyChangeRename

4.3 PROPOSED AUTOMATIC ADAPTATION

The important part of the OntoEvol system is to find the dependent entities that are affected by the DW schema change and adapt them accordingly. As DW involves a complex structure, there are different entities available which are dependent on each other. Hence, changes in a particular entity may affect others. One such entity is the *mapping* that exists between the data source and DW schema, through which the DW is populated.

When data source schema modifies its structure the DW schema also gets updated, thus, the mapping between them becomes invalid. Queries and views which are based on the DW schema are the other entities which become invalid due to

changes. In order to analyze the impact of a recent change and to maintain the DW system in a consistent state, it is necessary to adapt these entities.

The proposed system maintains a mapping between the DSO and DWO. When the mapping becomes invalid due to recent changes the system performs mapping adjustments between them. The queries which worked over the previous schema may not work for the new DW schema. Hence, a query rewriting is performed. Finally, the views maintained for the schema also becomes invalid, hence the views definitions are updated automatically. For each of the dependent entities discussed, the cost of updating them is computed. Based on the adaptation cost, the DW administrator can make an analysis of the impact of different changes over the DW schema and create the physical schema as required.

4.3.1 Mapping Adjustments

The *UpdateMapping* Algorithm given in Figure 4.10 has been proposed to perform the mapping adjustments. The changed concepts are obtained for DSO and DWO from the log entries. ChAO which acts as a log is used for storing the changes happening in the ontologies. Mappings are then established only for the changed resources. The previous mappings between these two ontologies are updated at the completion of the algorithm. DSO, DWO, mapping file and changed concepts are given as input to the algorithm. It produces the updated mapping file together with the number of entities affected and corrected as the output.

In the algorithm, first the changed concepts are obtained from log and read into CH for DSO and DWO (steps 1-6). If the type of change is *addition*, the similarities between the changed resources are computed for DSO and DWO (steps 6-10). If the change type is *deletion*, the concepts are searched in the mapping file and the corresponding mapping is removed (steps 11-13).

For a renamed concept the mapping entity is obtained and the concepts are renamed using information from the log, i.e., the old concept name and its mapping are deleted and the new concept name along with the mapping is added (steps 14-16). Finally the mapping file is updated with new mapping information. The total no. of entities affected and corrected is computed.

Algorithm UpdateMapping
Input: Ontologies DSO and DWO for mapping reconciliation, Ontology change information from ChAO of both ontologies, i.e., CH1 for DSO and CH2 for DWO.
Output: Updated Mapping, Number of mappings affected and corrected.

```
1    if CH ∩ CH. DSO.ChAO.NewChange then
2        CH1 = CH.ChAO
3    end if
4    if CH ∩ CH.DWO.ChAO.NewChange then
5        CH2 = CH.ChAO
6    end if
7    if ChAO.NewChange.ChangeType = ADDITION  then
8        NewMap ←Similarity(CH1,CH2)
9        Execute.update(MappingsFile, NewMap)
10       Count=Count+1
11   else if ChAO.NewChange.ChangeType = DELETION  then
12       Execute.update(MappingsFile, DeleteMap(CH1, CH2))
13       Count=Count+1
14   else if ChAO.NewChange.ChangeType = RENAME  then
15       Execute.update(MappingsFile, RenameMap(CH1_Old, CH2_Old,
                                                CH1_New, CH2_New))
16       Count=Count+1
17   else
18       Print("No Change")
19   end if
20   end if
21   end if
```

Figure 4.10 Algorithm UpdateMapping

4.3.2 Query Rewriting

As the DW schema has evolved the queries imposed previously need to be rewritten to work over the new DW schema. The proposed *QueryRewriting* algorithm given in Figure 4.11 is used by the OntoEvol to rewrite the queries. From the DWO, for a particular change, the concept changed, concept type and change type are retrieved. These are given as input to the algorithm. It produces the rewritten query, number of queries affected and corrected as the output.

If the concept type is a class and change type is *addition*, then the concept changed (fact, dimension or level) is updated in the *FROM* clause of the query (Steps 1-5). If the concept type is a class and change type is *deletion* or *rename*, then the *FROM* clause of the query is modified accordingly (Steps 6-9). If the concept type is a data property and change type is *addition*, then the concept changed (fact property, dimension property or level property) is updated in *select*, *where*, *groupby* or *orderby* clause of the query (Steps 14-17). If the concept type is a data property and change type is *deletion* or rename, then the concept changed (fact property, dimension property or level property) is updated in *select*, *where*, *groupby* or *orderby* clause of the query (Steps 15-22).

4.3.3 View Rewriting

A view is a virtual table in the DW defined by a query. It is mainly used when data security is required and when data redundancy is to be kept to the minimum while maintaining data security. When the underlying data source and DW changes its schema structure the views may become invalid. Hence, one important issue is to maintain the view consistency upon any structural changes. In order to find the number views affected and to rewrite the view definition, the steps given for query rewriting is applied. The changes are then automatically propagated to the DWO.

```
QueryRewrite(DWO, ConceptChanged, Concept Type, ChangeType,QueryWorkload)
1    if Concept Type ==Class then
2        c = ConceptChanged
3        if  ChangeType = =Addition then
4            Get suggestion from user
5            RewriteQuery in FROM clause with c
6        else if  (ChangeType = = Deletion | ChangeType = = Rename) then
7            RewriteQuery in FROM clause with c
8        end if
9        end if
10       Count++
11   else if Concept Type ==DataProperty then
12       dp = ConceptChanged
13       d= Domain(dp)
14       if  ChangeType = =Addition then
15           Get suggestion from user
16           SearchQuery for d
17           RewriteQuery in SELECT |WHERE |GROUPBY | ORDERBY clause
             with dp
18       else if  (ChangeType = = Deletion | ChangeType = = Rename) then
19           SearchQuery for d
20           RewriteQuery in SELECT |WHERE |GROUPBY clause with dp
21       end if
22       end if
23       Count++
24   end if
25   end if
```

Figure 4.11 Algorithm QueryRewrite

4.3.4 Impact Analysis

The impact of a particular set of changes imposed on the DW schema is obtained by computing the adaptation cost of the dependent entities. In the proposed approach the mapping, queries and views are adapted automatically. The automatic adaptation cost is given as a sum of number of changes propagated on the DW schema and cost of manually discovering and adjusting entities that escaped the automation. The details of the adaptation cost have been discussed in section 4.5.2. Based on the adaptation cost the DWA may decide to construct the DW schema at the physical level.

4.3.5 Data Warehouse Schema Construction

The final step of OntoEvol system involves in the transformation of DWO to physical schema in the underlying database. For the physical schema construction of the DW the steps defined in section 3.2.7 of Chapter 3 are considered.

Thus, the *Schema Evolution* and *Automatic adaptation* part of the proposed OntoEvol approach effectively propagate and adapt to the given changes in requirements or the data source. For the illustration and evaluation of the proposed OntoEvol approach the TPC-H case study has been used, which is explained in the following section.

4.4 CASE STUDY: TPC-H

The details of TPC-H benchmark has been discussed in section 3.4 of Chapter 3. The various steps involved in the OntoEvol approach explained in section 4.3 are applied to the case study. Using the results obtained the proposed approach is evaluated, which has been given in the next section.

For the case study, the data source schema, DW requirements and the DW schema represented in ontology format are taken as inputs. For data source schema and DW requirements, the DSO and DWRO based on TPC-H benchmark discussed

in section 3.2.1 of Chapter 3 are considered. The DW schema is taken from the DWO which has been derived from the proposed OntoMD approach discussed in section 3.2.5 of Chapter 3. Any change that happened in the DSO or DWRO are retrieved from the log and represented as change set. If the change type is *addition*, for individual change in the change set, its multidimensional type is identified. Based on the changes over DWO, any dependent entities affected are automatically adapted by the OntoEvol approach. Following are the steps applied for the case study:

1. Representation of Inputs
2. Change Identification
3. Change Propagation
4. Adaptation of dependent entities

4.4.1 Representation of Inputs

Once the inputs, data source schema, DW requirements and DW schema are available in ontology format, the proposed evolution system begins by considering a recent change either in the DSO or DWRO. For example, in the TPC-H domain, to increase the profit for the items shipped, a new *decision goal* "Increase Revenue through promotions" has been added in the requirements and its *information goal* is identified as "Study Revenue by Customer Promotions" and *information requirements* as "Analyze Revenue based on Customer for a given Promotions". The *context* for the new *information requirement* is "Customer" and "Promotions". The *measures* for analyzing Revenue are "ExtendedPrice" and "Discount". This new requirement is added to the DWRO which is shown in Figure 4.12.

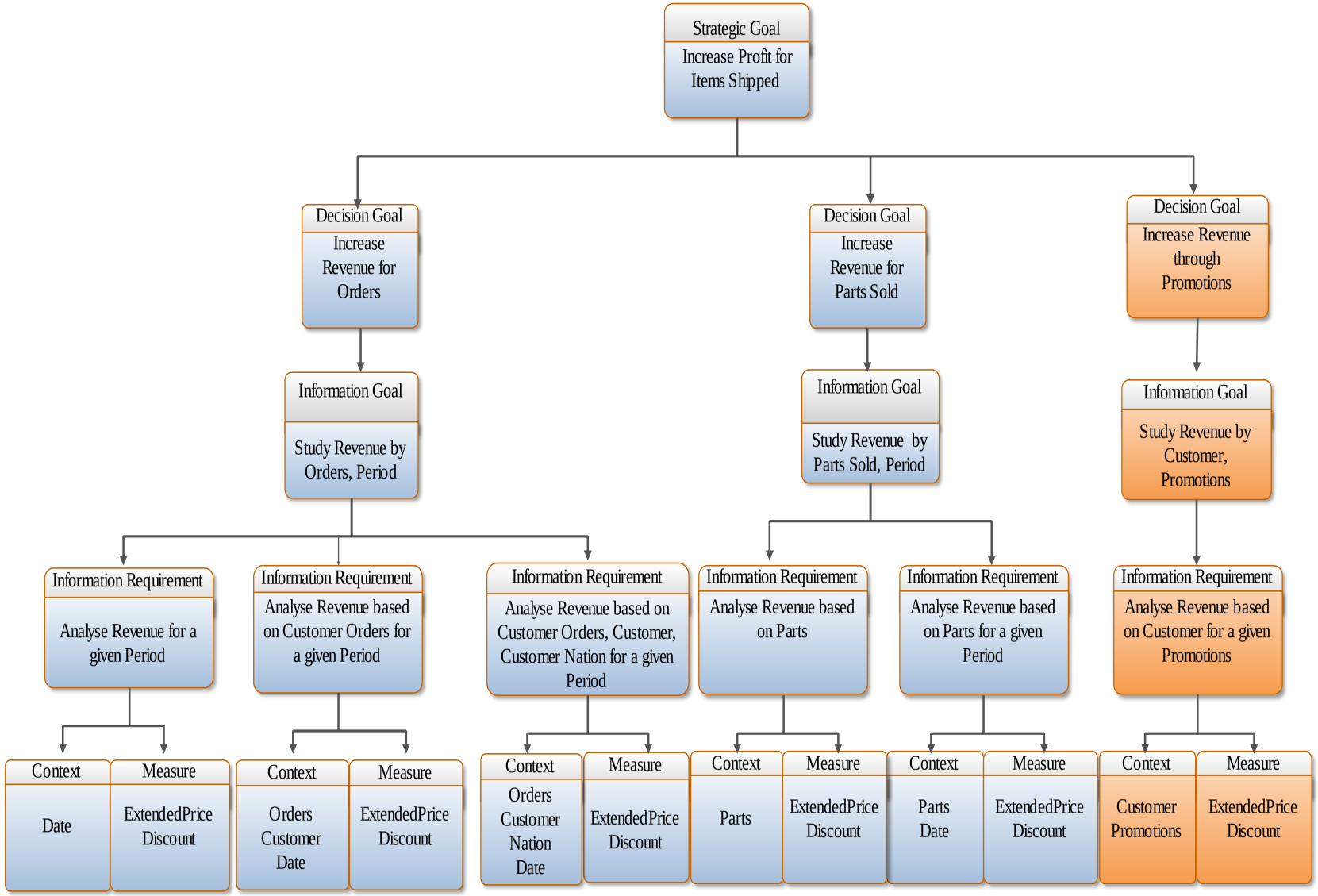

Figure 4.12 Data Warehouse Requirement

As new requirement has been included for the business analysis, the data source is verified for the existence of this information. In addition to the above changes in the requirements, the other change that has occurred over the DSO is given in Table 4.2.

Table 4.2 Change Set

Data Source Change	Data Source Ontology Change	Entity Changed
ADDITION		
Table	Class	Promotion
Attribute	Data Property	Promotion _p_id
Attribute	Data Property	Promotion _p_name
Attribute	Data Property	Promotion _p_category
Attribute	Data Property	Promotion _p_subcategory
Attribute	Data Property	Promotion _p_cost
Attribute	Data Property	Promotion _p_begdate
Attribute	Data Property	Promotion _p_enddate
Attribute	Data Property	Promotion _p_total
RENAME		
Attribute	Data Property	OldName:Customer_c_comment , NewName:Customer_c_feedback
Attribute	Data Property	OldName:Part_p_category, NewName: Part_p_model
DELETION		
Attribute	Data Property	Customer_c_mktsegment
Attribute	Data Property	Part_p_container

4.4.2 Change Identification

This step identifies the multidimensional type when the change type is *addition*. The algorithms *FindClassType* and *FindDPType* explained in section 4.2.4 are applied to derive the multidimensional type for the class “Promotions” and its data properties. As the class “Promotions” added to DSO has 1:n relationship with existing fact “LineItem”, it is identified as a dimension. Thus, the properties of the class “Promotions” becomes the descriptive data properties of the dimension “Promotions” according to algorithm *FindDPType*. Table 4.3 gives the details of multidimensional type for the added entities that are to be propagated to the DWO.

Table 4.3 Multidimensional Type

Data Source Ontology Change	Entity Changed	Multidimensional Type
Class	Promotions	Dimension Class
Data Property	Promotion _p_id	Dimension Data Property
Data Property	Promotion _p_name	Dimension Data Property
Data Property	Promotion _p_category	Dimension Data Property
Data Property	Promotion _p_subcategory	Dimension Data Property
Data Property	Promotion _p_cost	Dimension Data Property
Data Property	Promotion _p_begdate	Dimension Data Property
Data Property	Promotion _p_enddate	Dimension Data Property
Data Property	Promotion _p_total	Dimension Data Property

Table 4.4 DWO Changed Concepts

DW Ontology Change	Change Applied		
ADDITION			
Dimension Class	Add Promotions	Add Domain:	Add Range:
Dimension Data Property	Add Promotion _p_id	Add Domain: Promotion	Add Range: Integer
Dimension Data Property	Add Promotion _p_name	Add Domain: Promotion	Add Range: String
Dimension Data Property	Add Promotion _p_category	Add Domain: Promotion	Add Range: String
Dimension Data Property	Add Promotion _p_subcategory	Add Domain: Promotion	Add Range: String
Dimension Data Property	Add Promotion _p_cost	Add Domain: Promotion	Add Range: Double
Dimension Data Property	Add Promotion _p_begdate	Add Domain: Promotion	Add Range: Date
Dimension Data Property	Add Promotion _p_enddate	Add Domain: Promotion	Add Range: Date
Dimension Data Property	Add Promotion _p_total	Add Domain: Promotion	Add Range: Double
RENAME			
Dimension Data Property	Delete Name:Customer_c_comment , Add Name:Customer_c_feedback	-	-
Dimension Data Property	Delete Name:Part_p_category, Add Name: Part_p_model	-	-
DELETION			
Dimension Data Property	Customer_c_mktsegment	Delete Domain: Customer	Delete Range: String
Dimension Data Property	Part_p_container	Delete Domain: Part	Delete Range: String

4.4.3 Change Propagation

Table 4.4 presents the changes that are propagated over the DWO. For adding the dimension class “Promotions”, its domain, range and data properties are

added. For adding a dimension data property, for example, “Promotion p_id”, its corresponding domain and range are included in the DWO. For renaming the data property of Customer dimension, for example, “Customer_c_comment”, its old name is deleted and new name is added as “Customer_c_feedback”. And, for deleting the data property of Customer dimension, for example, “Customer_c_mktsegment”, its domain and range are deleted from the DWO. Figure 4.13 represents the final DWO after the changes are applied.

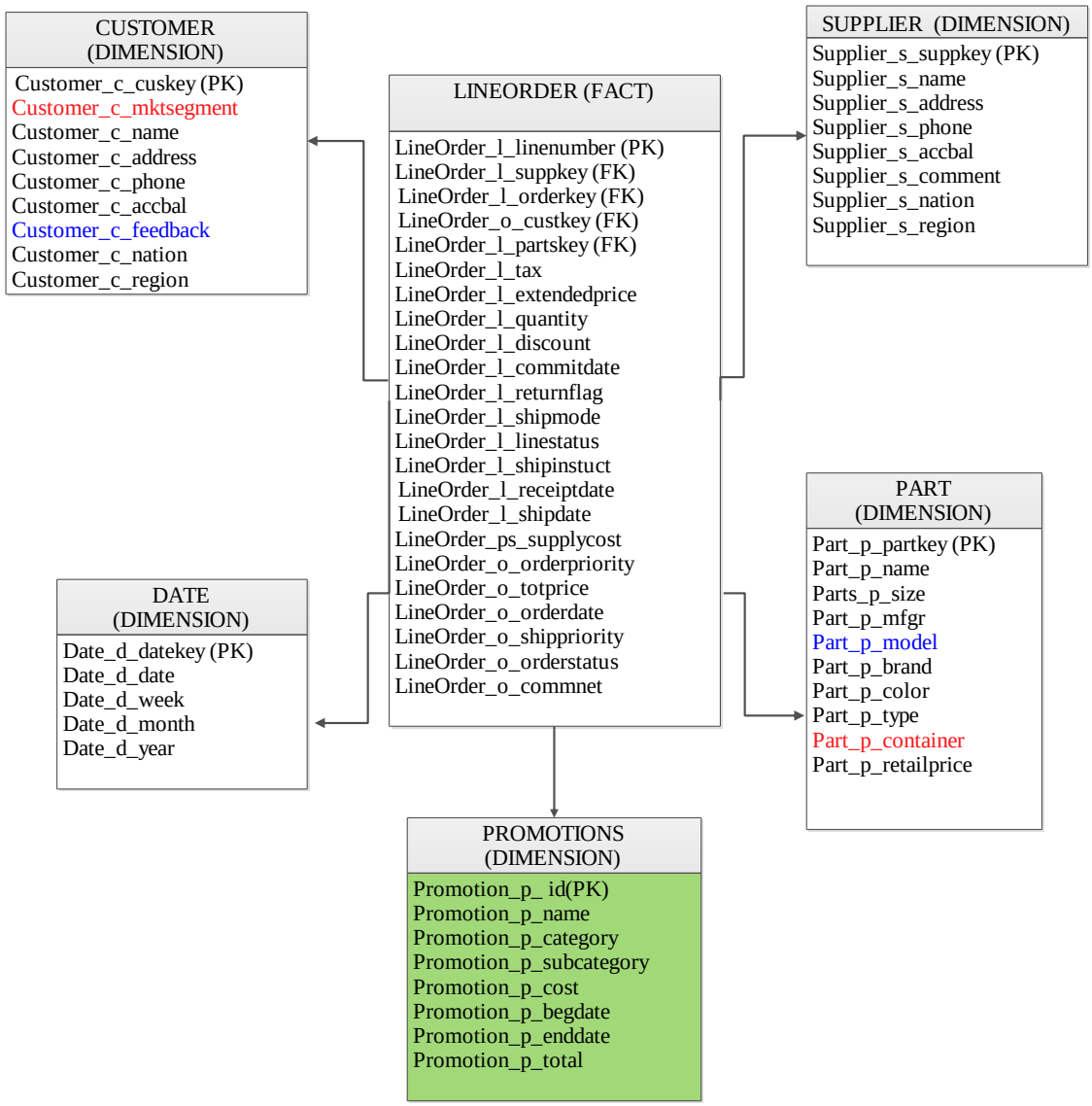

Figure 4.13 Data Warehouse Schema Ontology

4.4.4 Adaptation of dependent entities

This step involves in adapting the dependent entities, after the changes in the change set given in Table 4.2 has been successfully propagated over DWO. The Customer and Part dimension classes are affected by the changes and a new dimension class Promotions has been included in DWO.

i. Mapping Adjustments

In order to update the mapping between DSO and DWO, the proposed algorithm *UpdataMapping* is used by the OntoEvol approach. Figure 4.14 represents the mapping rewritten for the Customer dimension class. As the data property *Mktsegment* has been removed from DSO and from DWO, the corresponding mappings between these concepts are removed from the mapping document. The data property *Comment* in the DSO is renamed as *Feedback* which has been updated in the DWO. Hence, the mapping between *Comment* from DSO to DWO is deleted and a new mapping for the data property *Feedback* is added between DSO to DWO. Similar steps are applied for other changes involved in the change set.

ii. Query and View Rewriting

The SSB (O'Neil *et. al.,* 2007) queries and views are used in order to verify query and view rewriting performed by the OntoEvol approach. For example, in the change set given in Table 4.2, the data property *p_category* from Part dimension class is renamed as *p_model*. Using the *QueryRewrite* Algorithm the *select*, *where*, *groupby* and *orderby* clause in the queries are searched. As the Query 2.1, Query 4.2 and Query 4.3 contain the data property *p_category* it is renamed as *p_model*. Similarly, View 4 and View 5 are affected by the addition, deletion and rename operations given in the change set.

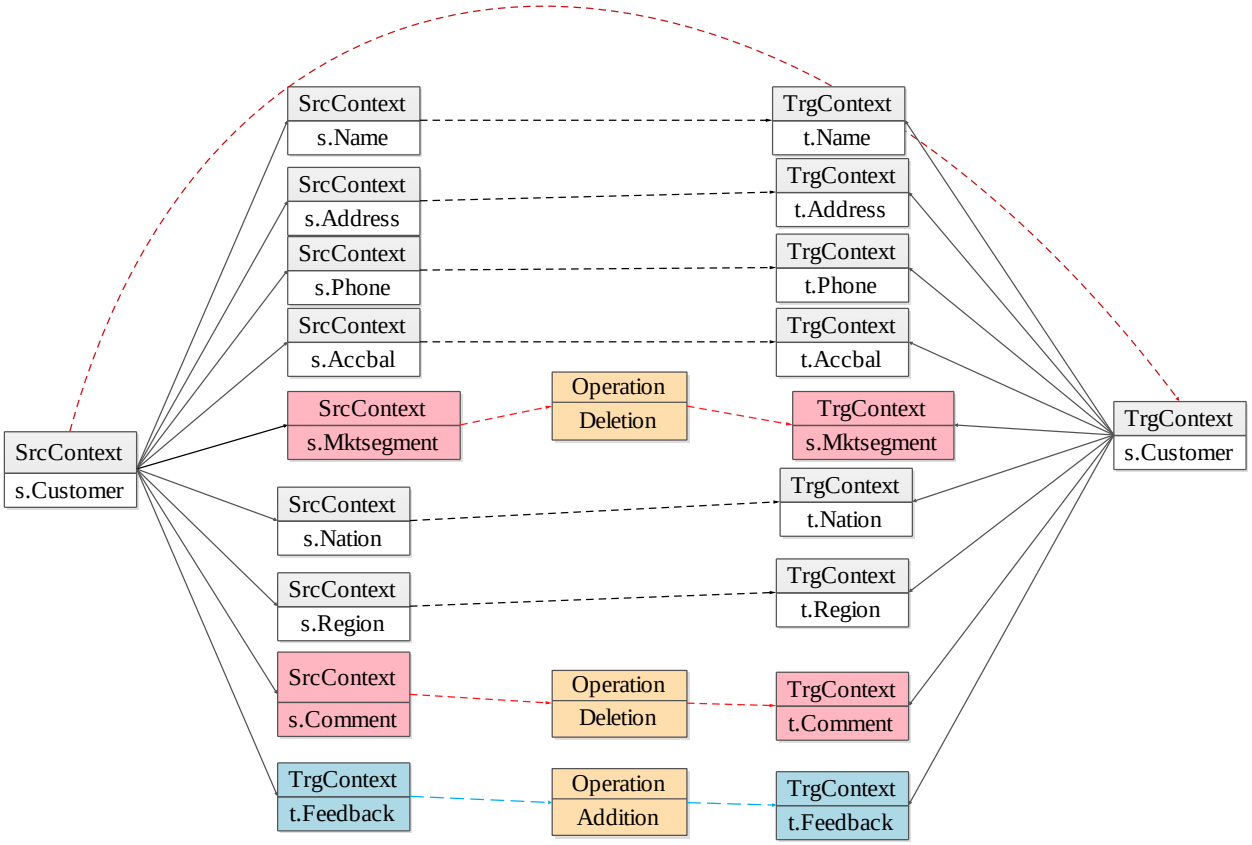

Figure 4.14 Mapping Adjustments

Figure 4.15 and Figure 4.16 represents a sample query and view rewriting that has been executed by the proposed adaptation system. In Query 4.3 shown in Figure 4.15, the *p_container* attribute has been removed, the *p_category* attribute has been renamed as *p_model* and a new attribute *pr_category* of Promotions table has been added. In View 5, shown in Figure 4.16, the *p_category* attribute has been renamed as *p_model*. In Table 4.5 the details about the query and view affected and corrected for individual change has been given.

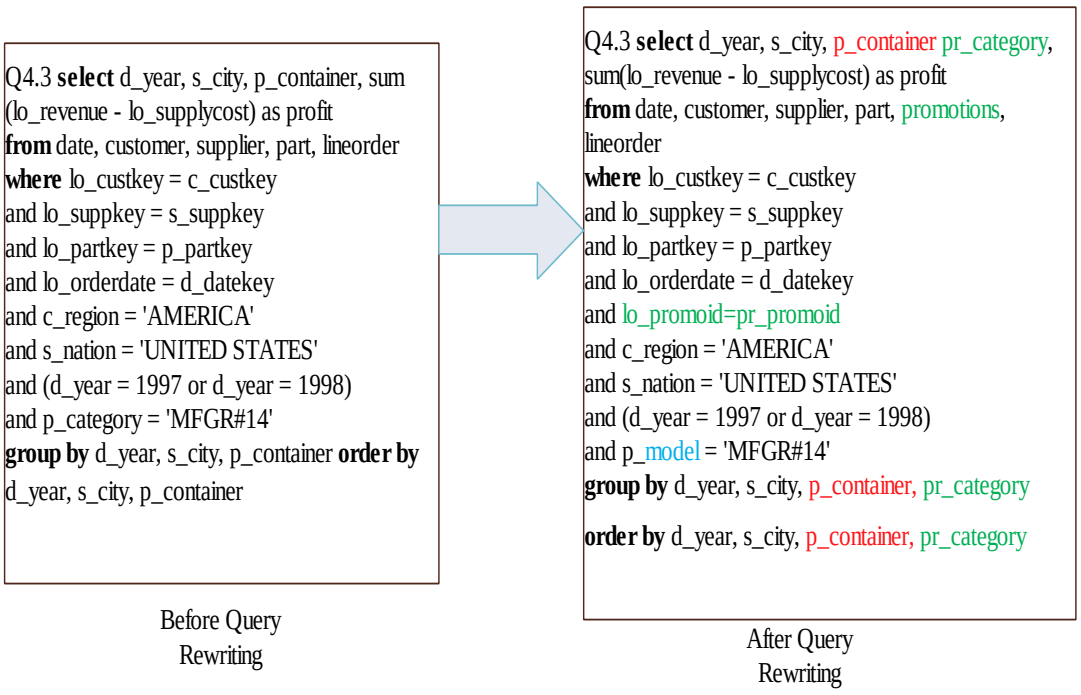

Figure 4.15 Query Rewriting

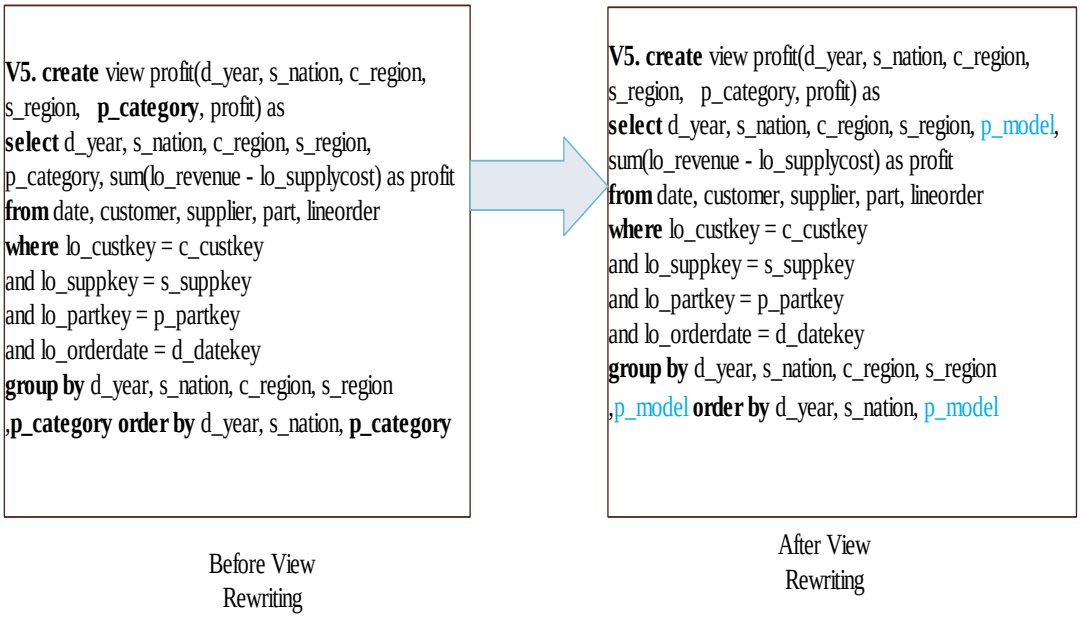

Figure 4.16 View Rewriting

Table 4.5 Query and View Adaptation

Data Source Ontology	Entity Changed	Query Affected and Corrected	View Affected and Corrected
ADDITION			
Class	Promotion	Q4.1, Q4.2 and	V4,V5
Data Property	Promotion _pr_id	Q4.1, Q4.2 and	-
Data Property	Promotion _pr_name		-
Data Property	Promotion _pr_category	Q4.1, Q4.2 and	V4,V5
Data Property	Promotion _pr_subcategory	-	-
Data Property	Promotion _pr_cost	-	V4,V5
Data Property	Promotion _pr_begdate	-	
Data Property	Promotion _pr_enddate	-	
Data Property	Promotion _pr_total	-	V4,V5
RENAME			
Data Property	OldName:Customer_c_comment ,	-	-
Data Property	OldName:Part_p_category, NewName: Part_p_model	Q2.1, Q4.2 and Q4.3	V4,V5
DELETION			
Data Property	Customer_c_mktsegment	Q4.2	-
Data Property	Part_p_container	Q4.3	-

4.5 RESULTS AND DISCUSSION

In this section the experimental setup for the evaluation of the proposed approach is provided. The evaluation is carried out in terms of its effectiveness to correctly propagate the changes over the DW schema and its efficiency to adapt the dependent entities. Further, a comparative analysis of the proposed OntoEvol approach and existing approaches has been given.

4.5.1 Experimental Setup

In order to analyze the effectiveness, different change set is considered by the proposed approach. Each change set consists of elementary changes. One such change set is given in Table 4.2 and its propagation over the DWO has been explained in the case study. The set of evolution operations or changes that are

considered in the TPC-H source schema includes, addition of attributes and table, renaming of attributes and table, and, deletion of attributes and table. A total of 143 evolution operations were encountered and the distribution of occurrence per kind of operation is shown in Figure 4.17.

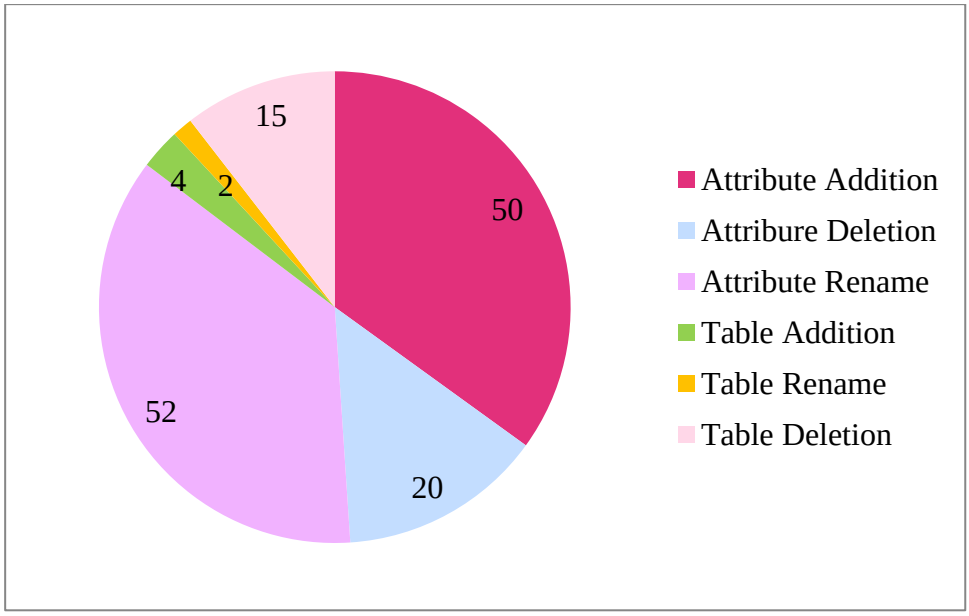

Figure 4.17 Distribution of occurrence per kind of evolution operations

In Table 4.6, the results are summarized for different kinds of events. It is observed that most of the activities are affected by attribute additions and renaming, since these kinds of operations are the most common in the given scenario. Most important, it is observed that the proposed evolution approach can effectively adapt activities to the examined kinds of operations.

Table 4.6 Affected and Corrected Operations

Evolution Event Type	Total Affected	Total Corrected
Attribute Addition	50	49
Attribute Deletion	20	20
Attribute Rename	52	51
Table Addition	4	4
Table Deletion	2	2
Table Rename	15	14

4.5.2 Effectiveness Analysis

The effectiveness of OntoEvol approach is derived using the following metrics:

$$Effectiveness = \frac{Number\ of\ Entities\ Corrected}{Number\ of\ Entities\ Affected}$$

Figure 4.18 shows the comparison of no. of attributes affected and that are corrected by using the proposed approach. Here, it is observed that the system achieved 98% effectiveness for attribute addition, 100% for attribute deletion and 98% for attribute rename.

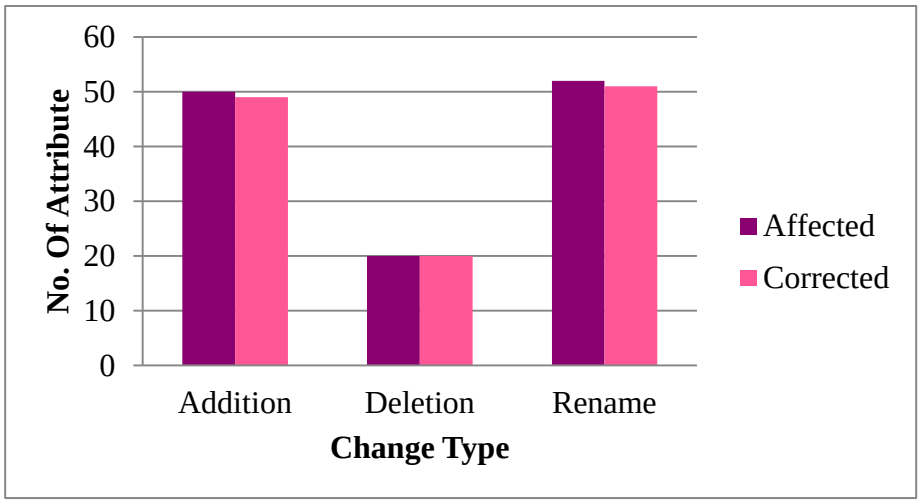

Figure 4.18 No. of Attributes Affected and Corrected Status

The comparison of no. of tables affected and corrected by using the proposed approach is given in Figure 4.19. From the figure it is observed that the system achieved 100% effectiveness for table addition, 100% for table deletion and 93% for table rename.

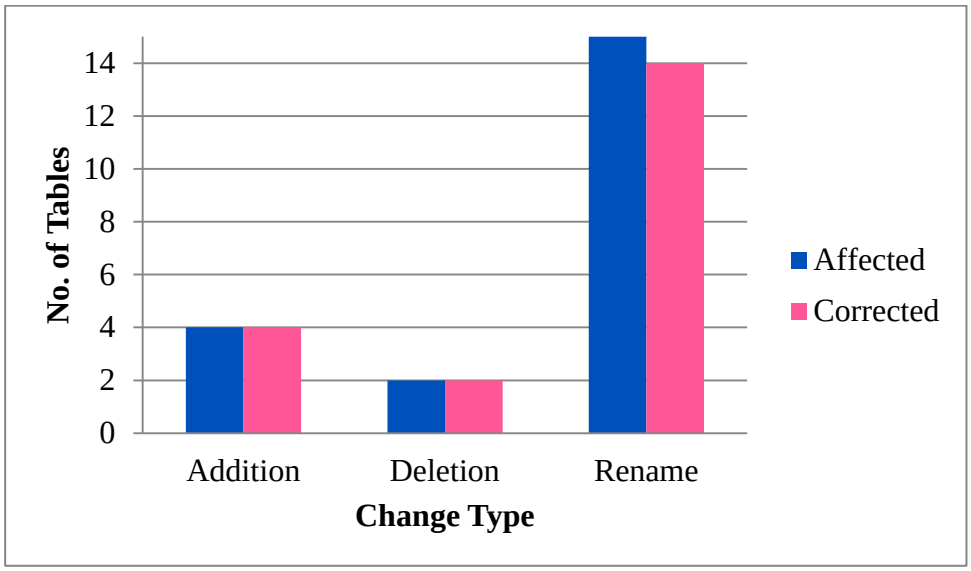

Figure 4.19 No. of Tables Affected and Corrected Status

Figure 4.20 shows the comparison of the proposed approach with existing evolution approaches in terms of its effectiveness to correctly propagate the changes over the DW schema. From the Figure 4.20, it is observed that the OntoEvol approach achieved 98% of effectiveness when compared to existing DWE (Solodovnikova and Niedrite, 2011) and MVTDW (Oueslati and Akaichi *et al.*, 2011) approaches which produced 80% and 85% effectiveness respectively, for the evolution task.

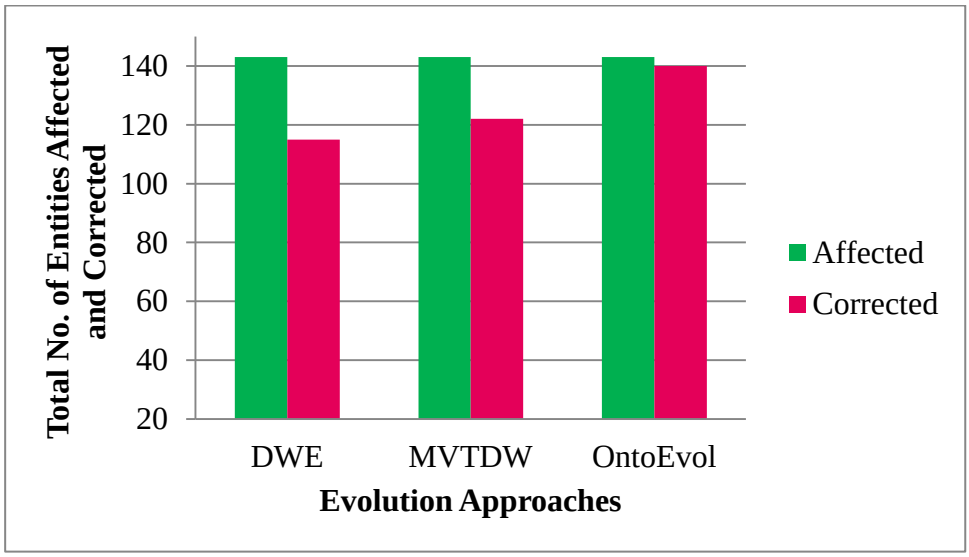

Figure 4.20 Comparison of Different Evolution Approaches

4.5.2 Efficiency Analysis

In order to evaluate the efficiency of the proposed OntoEvol approach the cost of manually adapting the dependent entities for different change set is compared with the proposed ontological approach. The manual effort comprises of detection, inspection and where necessary the rewriting of affected entities by an event.

Human effort for manual handling of an entity e, for a change c, is expressed as:

$$MC_e^c = AX_e^c + RX_e^c \quad \text{..... (4.1)}$$

Where,

AX = No. of Mapping/Query/View entity e, affected per change c, that is manually detected.

RX = No. of Mapping/Query/View entity e, affected per change c, which must be manually adjusted.

For a set of changes C, and a set of manually adapted entities E, the overall cost of manual adaptation is given as:

$$CMA = \sum_{c \in C} \sum_{e \in E} MC_e^c \quad \text{..... (4.2)}$$

Automatic handling of the dependent entities using the proposed ontological approach is quantified as a sum of no. of changes imposed on the DW schema CS, and cost of manually discovering and adjusting entities AMC that escape the automation E_d. The latter cost AMC is expressed as:

$$AMC = \sum_{c \in C} \sum_{e \in E_d} MC_e^c \quad \text{..... (4.3)}$$

Thus, the overall cost of automated adaptation is given by,

$$CAA = CS + AMC \quad \text{..... (4.4)}$$

For example, the Table 4.2 represents the sample change set CS1, whose adaptation cost has been given in Table 4.6 along with the adaptation cost for other change sets. The total no. of changes CS, propagated over the DW schema is 13 for CS1. For each change, i.e. evolution operation, the no. of mapping, query and view entities affected are computed. Here, TAX is total mapping affected, which is 13 and TRX is total mapping rewritten manually, which is 15 for the entire CS1. Hence, the total manual effort MC is computed as 28. Thus, the overall cost of manual adaptation, CMA is 28. In order to compute automatic adaptation cost by using the OntoEvol system, CS is added with the no. of entities that escaped automation E_d, which is 2 for the given mapping. Thus, the overall cost of automatic adaptation CAA for the given mapping is 15. Similarly, the values of CMA and CAA can be computed for query and view entities for the change set CS1. In order to compute the efficiency of the OntoEvol system, various change sets are imposed over the DW schema and the adaptation cost computed are given in Table 4.7.

Where,

TAX – Total no. of Mapping/Query/View entities affected per change c, by event an e, which is manually detected.

TRX – Total no. of Mapping/Query/View per change c, which must be manually re-written.

TMC – Total human effort for manual handling of schema evolution for a change c, over an entity e.

CS – No. of changes imposed on the DW schema.

AMC – Cost for automated handling of schema evolution for a change c, over an entity e.

CMA – The overall cost of manual adaptation to the change c, for an entity e.

CAA – The overall cost of automatic adaptation to the change c, for an entity e.

Table 4.7 Impact Analysis

Change Set	Entity	Manual and Automatic Adaptation Cost						
		TAX	TRX	TMC	CS	AMC	**CMA**	**CAA**
CS1	Mapping	13	15	28	13	2	**28**	**15**
	Query	14	14	28	13	3	**28**	**16**
	View	10	10	20	13	2	**20**	**15**
CS2	Mapping	15	19	34	15	2	**34**	**17**
	Query	12	12	24	15	0	**24**	**15**
	View	9	9	18	15	0	**18**	**15**
CS3	Mapping	18	20	40	14	2	**38**	**16**
	Query	12	12	24	14	2	**24**	**16**
	View	10	10	20	14	0	**20**	**14**
CS4	Mapping	16	20	36	15	0	**36**	**15**
	Query	13	13	26	15	0	**26**	**15**
	View	10	10	20	15	0	**20**	**15**
CS5	Mapping	14	16	30	12	4	**30**	**16**
	Query	10	10	20	12	0	**20**	**12**
	View	10	10	20	12	2	**20**	**14**
CS6	Mapping	20	20	40	17	2	**40**	**19**
	Query	14	14	28	17	3	**28**	**20**
	View	10	10	20	17	2	**20**	**10**
CS7	Mapping	15	17	32	13	0	**32**	**13**
	Query	12	12	24	13	0	**24**	**13**
	View	10	10	20	13	2	**20**	**15**
CS8	Mapping	18	20	40	17	2	**38**	**19**
	Query	10	10	20	17	0	**20**	**17**
	View	10	10	20	17	0	**20**	**17**
CS9	Mapping	12	12	24	12	0	**24**	**12**
	Query	8	8	16	12	0	**16**	**12**
	View	8	8	16	12	0	**16**	**12**
CS10	Mapping	14	16	30	15	2	**30**	**17**
	Query	9	9	18	15	0	**18**	**15**
	View	10	10	20	15	2	**20**	**17**

The following Figure 4.21 graphically represents the impact of evolution on mapping. A comparative analysis is made between the existing manual adaptation approach and automated adaptations using the proposed OntoEvol approach for various change sets. It has been observed that mapping adjustments using the ontological approach resulted in an average of 53% minimal cost compared to the manual adaptation.

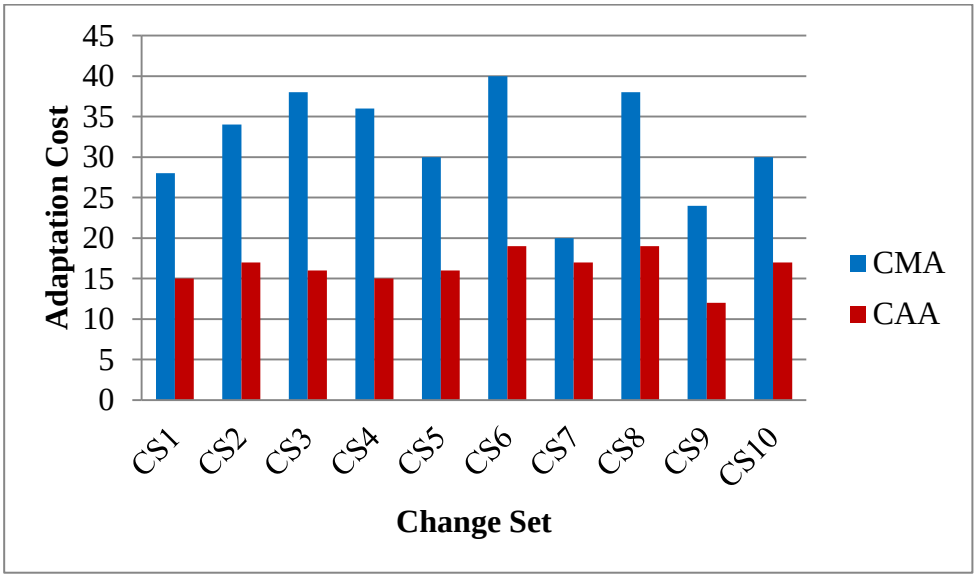

Figure 4.21 Comparision of Manual and Automated Adaptation Cost for Mapping

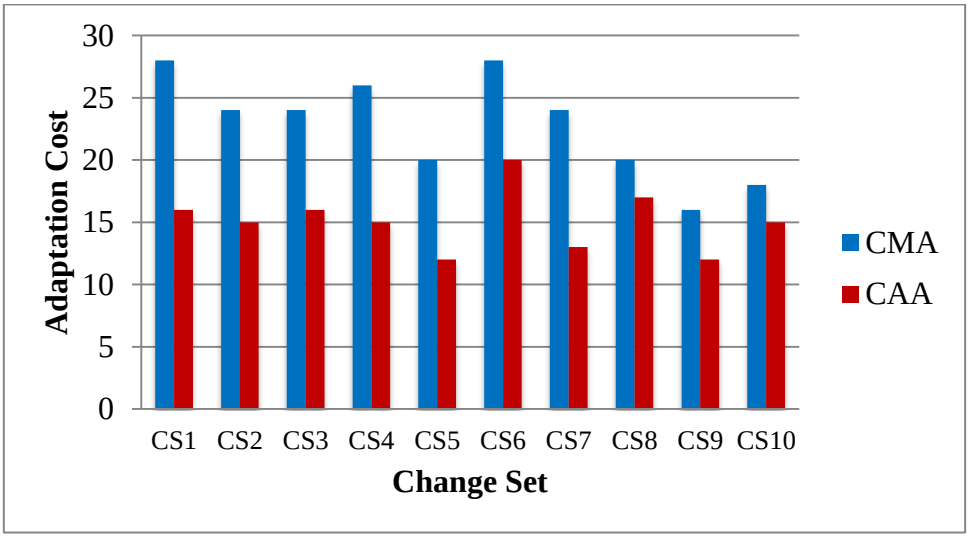

Figure 4.22 Comparision of Manual and Automated Adaptation Cost for Queries

The Figure 4.22 graphically represents the impact of evolution on queries with a comparison of manual adaptation cost and automated adaptation cost for various change sets. It has been observed that query rewriting using the ontological approach resulted in an average of 67% minimal cost compared to the existing manual adaptation.

The following Figure 4.23 graphically shows the impact of evolution on views with a comparative cost of manual adaptation approach and automated adaptation using the proposed ontological approach for various change sets. It has been observed that view rewriting using proposed approach resulted in an average of 74% minimal cost compared to the manual adaptation.

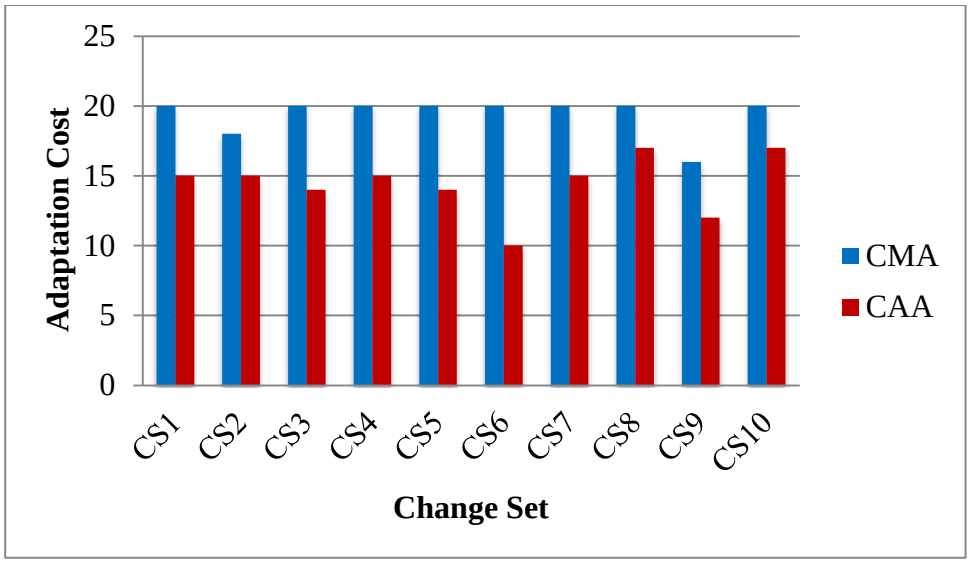

Figure 4.23 Comparision of Manual and Automated Adaptation Cost for Views

From the above Figures 4.21 to 4.23, it is found that the automated cost (CAA) of adaptation to Mapping, Query and View produced by the OntoEvol system is comparatively less than that of the manual cost of adaptation (CMA) used by the existing schema evolution approaches.

Using the obtained results it has been inferred that the proposed approach achieved better results when compared to the existing evolution approaches.

i. The proposed OntoEvol system achieved 13% better results in effectively propagating the given set of changes when compared to the existing MVTDW (Oueslati and Akaichi *et al.*, 2011) approach and 17% better than DWE (Solodovnikova and Niedrite, 2011) approach.

ii. In case of efficiency to adapt the dependent entities such as mapping, queries and view, for the given change sets, the proposed OntoEvol system provided 62% minimal cost when compared to the manual adaptation.

4.6 SUMMARY

The DW is considered as the core component of the modern decision support systems. As the information sources and business requirements from which the DW is derived, frequently change, it may have its impact on the DW schema. The existing works on DW evolution, such as schema versioning and schema evolution mainly concentrate on changing the schema structure at the physical level. The proposed approach handles evolution of the DW schema at the ontological level. The ontological representation of the data source, requirements and DW schema helps to provide automation of evolution task. The impact that the evolution has brought over the DW schema is analyzed. Based on the adaptation cost the DW designer is left with the choice of carrying the changes at the existing physical schema of the DW or create a new version of the schema from ontology.

Compared to existing approaches, the effectiveness of the proposed approach is better in handling the changes over the DW schema. Moreover, the proposed ontological approach provides minimal adaptation cost when compared to the manual adaptation of the dependent entities. Thus, current chapter focused on managing the schema when requirements or the data source evolved. Another issue in the DW design is to manage the schema for performance optimization which is the focus of the next chapter.

CHAPTER 5

OPTIMIZATION OF DATA WAREHOUSE SCHEMA PARTITIONING TECHNIQUES

5.1 INTRODUCTION

Partitioning is a design technique applied to the DW schema for dividing a single table into two or more partitions (fragments), thus improving query performance and optimizing resource utilization. As the DWA can manage only a limited set of fragments in the underlying database it is essential to select only optimal set of fragments during the design, which can reduce the overall query execution cost. The existing approaches on DW schema partitioning adopted evolutionary approaches such as genetic algorithm, as fragmentation selection is an optimization problem involving large search space (Boukhalfa *et al.,* 2009, Dimovski *et al.,* 2011 and Bellatreche, 2012).

Apart from fragment selection, there are several issues that exist during partitioning, which has not been much addressed in the literature. The dimension table selection is an important factor in order to referentially partition a fact table. Moreover, when the given business scenario contains a *big* dimension, involving large number of attributes, then the appropriate fragmentation technique needs to be applied to partition the dimension. As the fragmentation selection involves a complex problem of selecting optimal fragments from the given set of large fragments, the existing evolutionary algorithms need to be improved for better results. Further, the existing partitions need to be managed when the business analysis need changes, as the query imposed over the DW may evolve. The proposed approach on DW schema partitioning focus on solving the discussed issues.

This chapter presents the details of the proposed optimized referential partitioning (ORP) approach. Section 5.2 presents the different steps involved in ORP and discusses each of them in detail. To handle evolving queries the partition management of the ORP approach has been discussed in section 5.3. To apply ORP approach SSB case study has been used which is explained in section 5.4. The details of evaluation carried out for the proposed approach and the comparative study with existing methods have been discussed in section 5.5. Summary of the chapter has been provided in section 5.6.

5.2 ORP: PROPOSED OPTIMIZED REFERENTIAL PARTITIONING APPROACH

In this section the proposed ORP approach has been described in detail. A DW modeled as a star schema is considered for partitioning, which consists of set of D dimension tables and a fact table F, which need to be referentially partitioned in order to minimize the cost of a given set of queries Q. The DWA can maintain a maximum of N number of fragments in the underlying database. The proposed method offers the DW designer to partition a fact table with respect to the chosen dimension table(s). The steps involved in the proposed referential partitioning are given in Figure 5.1.

In order to automate the partitioning process, the ORP approach uses ontology representation of the DW schema. Using the DW schema ontology (DWO) and the query workload Q, the first step involves in the selection of appropriate dimension and its attributes. With the chosen dimension and its attributes a fragment schema is constructed. This fragment schema, having integer representation for the values of the attributes forms one of the possible solutions for fragmentation. From this initial solution several other solutions are generated, and the optimal set of fragments is obtained by the proposed hybrid algorithm. Based on optimal fragment schema the dimension table(s) is horizontally fragmented. The fact table is fragmented with reference to the fragments of dimension tables. For a DW schema involving big dimension, the table is vertically partitioned by applying attribute

clustering and for each vertical fragment generated, the above discussed horizontal partitioning is applied.

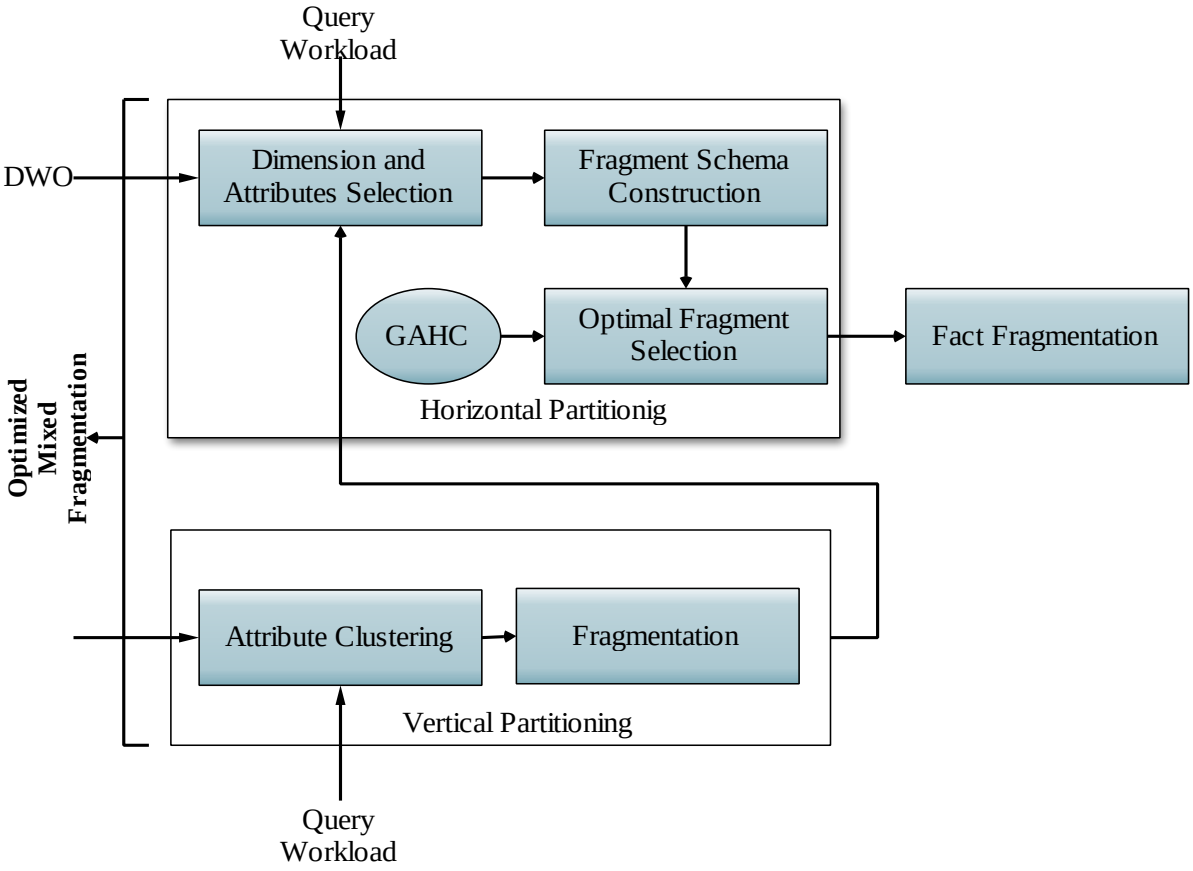

Figure 5.1 Proposed ORP Approach

5.2.1 Dimension Table and Attribute Selection

The DW schema may involve a large number of dimension tables. When the number of dimension table fragments is more, the number of fact fragments becomes large. Hence, the choice of dimension table and its attributes has an impact on the number of fragments that is generated. To select the appropriate number of dimension tables and attributes for partitioning, a *dim_selection* matrix and an *attr_selection* matrix are constructed in this step. The format of the *dim_selection* matrix is shown in Table 5.1. The rows of the matrix represent the dimension tables and the column represents the parameters which are used as the selection criteria for the dimensions. The three parameters used in the proposed approach are: *frequency, attributes* and *size.*

Table 5.1 Dim_Selection Matrix

Parameters / Dimension	Frequency	Attributes	Size
D1	n	n-3	n-1
D2	n-1	n	n
D3	n-2	n-2	n-3
Dn	n-3	n-1	n-2

The frequently used table when partitioned reduces the query execution time. Hence, the first parameter is the *frequency* of the dimension table, which is obtained from the query workload. The second parameter is the *attributes*, which represent the total number of attributes of a dimension table in the query workload Q. Choosing a dimension table with large number of attributes in the queries for partitioning, has an impact on the query execution. The third parameter is the *size* of the dimension table. Partitioning a large table would reduce the query execution time. Based on the values of each parameter, the individual dimension is assigned a score. Dimension having higher values for a parameter gets high score, i.e. n, which is the total number of dimensions available. These scores occupy the cells of the *dim_selection* matrix. The rows of the matrix are added to calculate the total score for a dimension. Using this matrix the DWA can select one or more dimensions having high scores for partitioning process.

The three parameter values explained above are obtained automatically using the proposed algorithm *DimSelection* which is given in Figure 5.2. The algorithm takes DWO representing the DW schema and the query workload Q as inputs. It produces the values of the three parameters for each dimension as the output. For each class in DWO, the algorithm first checks if it is a dimension class (Steps 1-2).

```
DimSelection(DWO, MDList, Q)
1 for all ci ∈ C do
2     if ci ∈ DimensionList then
3           //find table_frequency
4           for each qj in Q
5                 if ci in FROM clause then
6                       Fcount++;
7                 end if
8           end for
9           ci .frequency=Fcount;
10          //find attr_frequency
11          for all ci.dpi ∈ DP do
12                for each qj in Q
13                      if  ci.dpi in SELECT |WHERE |GROUPBY | ORDERBY
                                                              clause then
14                      Acount++;
15                      end if
16                end for
17          end for
18           ci .attributes=Acount;
19          // find table_size
20          Connect to DB
21          ci.size =Execute Query( select avg_row_len*num_rows from
                                    dba_tables)
22    end if
23 end for
```

Figure 5.2 Dim_Selection Algorithm

Next, for each of the dimension class the values of the parameters are obtained. In order to calculate the *frequency* value the dimension class name is searched in the *from* clause of each query belonging to the query workload Q, and its total count is obtained (Steps 3-9). To find the number of attributes, i.e. *attributes* value, each data property of the dimension class is searched in the *select, where, groupby and orderby* clause of the queries (Steps 10-18). The *size* of the dimension is obtained from the underlying database where the physical table is stored (Steps 19-21).

After the selection of dimension, the attributes for partitioning are derived by constructing the *attr_selection* matrix. For each of the dimension tables selected using *dim_selection* matrix, the *attr_selection* matrix is constructed as given in Table 5.2. The rows of the matrix represent the attributes of the chosen dimension and columns represent each query. If a particular attribute exists in the *select*, *where*, *groupby* or *oderby* clause of the queries, the cell value is represented as 1 or 0 otherwise. Using this matrix the DWA can choose one or more attribute present in the queries as partitioning attribute.

Table 5.2 Attribute_Selection Matrix

Queries / Attributes	Q1	Q2	Q3
A1	1	0	1
A2	0	1	0
A3	1	1	0
An	0	1	1

5.2.2 Fragment Schema Construction

Once the dimension table and its attributes are chosen for partitioning using the previous step, the next step involves the construction of the fragment schema. This schema represents the partition of the selected dimension table. The Table 5.3 shows the format of the fragment schema. The rows of the schema represent the attributes and columns represent the attribute's domain values. Based on the domain values of a particular attribute, the cells are filled with integer numbers from 1, 2, 3… N. The reason for using integer representation for the fragment schema is that, it is given as input for the proposed hybrid evolutionary algorithm to select optimal fragments.

Table 5.3 Fragment Schema

Values / Attributes	Domain		
A1	1	2	-
A2	1	2	3
A3	1	2	3
An	1	2	-

```
FragmentConstruction (DWO, Selected_Dim, Selected_attr)
1 for all ci ∈ C do
2       if ci ∈ Selected_Dim then
3               for all ci.dpi ∈ DP do
4                       if ci.dpi ∈ Selected_attr then
5                               rng := Rng(dpi);
6                               if (rng="Integer") | (rng="Date") then
7                                       Get predicates of dpi;
8                                       N=count(predicates);
9                               else
10                                      Get values of dpi from ti;
11                                      N=count(values);
12                              end if
13                              k=1;
14                              for m 1 to N
15                                      Fragment[n][m]=k++;
16                              end for
17                              n++;
18                      end if
19              end for
20      end if
21 end for
```

Figure 5.3 Fragment Construction Algorithm

The fragment schema is constructed using the proposed algorithm given in Figure 5.3. The DWO, list of dimensions and a list of attributes selected are given as input to the algorithm. It produces the fragment schema as the output. For each class in the DWO, if the class is the selected dimension, its data properties representing the attributes of the dimension are retrieved (Steps 1-3). For each of the data property, the range is obtained from the DWO (Steps 4-5). If the range value is "integer" or "date" then the predicates of the data property (attribute) is obtained from the user (Steps 6-8). A predicate represents a pure boolean expression over the attributes of a relation and constants of an attribute's domain. For other range values such as "string", the values are obtained from its corresponding table in the underlying database (Steps 9-12). The total number of values of the data property is computed. Based on the total count, the cells of the fragment schema are represented with integer numbers (Steps 13-17).

5.2.3 Optimal Fragment Selection

Depending on the attributes chosen for partitioning the number of partitions or fragments might be very large. If m_i is the number of fragments of the dimension table D_i, and g is the total number of dimension tables fragmented, then the total number of fragments N of the fact table is:

$$N = \prod_{i=1}^{g} m_i$$

As the DWA can manage only a limited set of fragments in the underlying database it is essential to select only optimal set of fragments which can reduce the overall query execution cost. The cost of a query is computed based on the number of disk I/Os. As fragmentation selection is an optimization problem involving large search space the existing works adopted heuristic approaches such as genetic algorithm and hill climbing algorithm. Hill climbing is an optimization technique which belongs to the family of local search (Selman et al., 2006). Genetic algorithms (GAs) are a general methodology for searching a discrete solution space in a way that is similar to the process of natural selection procedure in biological systems

(Mitchell 1998). Hill climbing search that uses only one solution can easily miss some promising areas of the search space, and thus it may get stuck in a local optimum. And genetic algorithms show lower solution quality with increasing problem size. In the proposed work on referential partitioning a hybrid genetic and hill climbing algorithm has been proposed in order to overcome the limitation of individual algorithms and improve the final solution.

```
1   Initialize the parameters:
2       population_size, max_generation;
3   Generate initial population P randomly with generation=1;
4   Apply HillClimbing for initial population
5   while generation <= max_generation do
6       Create new population P1;
7       Use a fitness function F to evaluate each individual in P;
8       for i=1 to population_size
9           Select two parents from P;
10          Perform crossover;
11          Perform mutation;
12          Place the new offspring into P1;
13      end for
14      for i=1 population_size
15          Apply HillClimbing for new population P1
16          Compute the fitness value for the new population P1
17      end for
18      Merge new offspring from P1 with old offspring from P
19      Select the fittest offspring and pace into P1 such that Size(P1) =N
20      P= P1;
21      generation =generation + 1;
22  end while
```

Figure 5.4 GAHC Algorithm

The proposed approach combines the principles of genetic and hill climbing algorithms where genetic algorithm is efficient at finding the best solution patterns and hill climbing is exploited to quickly tune solutions to reach local optimum. The steps of hybrid GAHC (genetic algorithm with hill climbing) are given in Figure 5.4.

At initiation the GAHC algorithm creates a set of random valid individuals (fragment schema) called initial population. The hill climbing algorithm starts with an individual chosen randomly from the initial population. It then attempts to find a better solution by incrementally changing (by step size) a single element of the individual. The original solution is replaced with the resultant if it has better fitness and then the step size is doubled. But, when the resultant does not have better fitness the step size is halved. Now, when the individual is at a local optimum the stepping begins again on the next individual. Upon reaching a local optimum for the individual, a new solution is randomly chosen from the initial population and hill climbing begins again. For every iteration the best prior solution is remembered.

From the obtained initial population the genetic principles are applied. First, the fitness of an individual (fragment schema) is measured using a cost function. Individuals (fragment schemas) are selected from the initial population for the crossover operation based upon their fitness values. The crossover occurs by mixing the two solutions together to produce two new individuals (fragment schemas). Next, the individual (fragment schema) is allowed to mutate in each generation, which changes the individual. The proposed algorithm continues with the process of several hill climbing iterations followed by genetic-principles iteration. After a certain number of iterations, the algorithm converges to a set of solutions to the problem at hand. Once the population has converged and no more offspring (fragment schema) produced is noticeably different from those in previous generations the algorithm terminates. The proposed GAHC algorithm applied for fragmentation selection has been explained in the case study given in section 5.4.3.

The quality of a solution is given by the fitness value is calculated by the Query cost. The cost for each fragment schema (estimating the number of inputs and outputs required for executing the set of queries) are computed using the cost model:

Query Cost = {(Size of each fact fragment) * (Length of each instance of fact)}
/ Page size of the disk. ……(5.1)

5.2.4 Optimized Mixed Fragmentation for Big Dimension

The previous steps described ORP approach with horizontal fragmentation (ORP-H) for partitioning fact table with reference to the chosen dimension table. In certain scenario the dimension table may be wider, i.e., it may involve a large set of attributes which is called a *big* dimension (Costa and Madeira 2004). Hence, applying horizontal partitioning alone might not be effective. In the proposed work, optimized mixed (hybrid) fragmentation (ORP-M) approach has been developed to solve the big dimension problem. In the mixed fragmentation the *big* dimension is fragmented vertically based on the columns (attributes), followed by a horizontal fragmentation on each vertical fragment.

A vertical fragmentation of a relation R produces fragments F1, F2, . . . , Fn each of which contains a subset of R's attributes. Compared to horizontal fragmentation, vertical fragmentation is inherently more complicated. When the horizontal partition consisting of n simple predicates the possible minterms is 2^n and some of them can be ruled out by existing constraints. Whereas, in vertical partitioning for m non-primary key attributes, the number of possible fragments is equal to B(k) (= the k_{th} Bell number), i.e., the number of partitions of a set with m members. For example B(15) = 10^9 (Ozsu and Valduriez 1999). In vertical fragmentation attributes usually accessed together are placed in one fragment and hence there is a need for some measure that would define more precisely how closely the attributes are related. Hence, in the proposed approach the information about the attributes are collected in the Query Attribute Matrix (QAM).

For the given query workload Q which is executed over relation R consisting of attributes (A_1, . . . , A_n) the QAM denotes which query uses which of the attributes. The rows represent workload queries and columns represent attributes from Q. In general terms QAM (i, j) is set to one if Q_i includes attribute A_i and to zero otherwise as shown in Table 5.4.

Table 5.4 Query Attribute Matrix (QAM)

Attributes / Queries	A1	A2	A3	An
Q1	1	0	0	0
Q2	0	1	0	0
Q3	0	1	1	0
Qn	0	1	0	1

The next step is to derive fragments that optimize data access for a given set of queries. As vertical fragments are built from attributes it is necessary to cluster the attributes with respect to the query workload. Following steps are used to cluster the attributes:

1. Find the attributes set W contained in the same queries from QAM.
2. Add the primary key of the table to each set.
3. Remove the duplicate sets.
4. Construct the execution tree:
    i. Include unused attributes and primary key of the table as root.
    ii. Extend tree by adding possible sets.
    iii. Repeat steps i and ii, until all attributes are included in one leaf.
5. Each leaf of the tree represents a valid candidate attribute clustering solution.
6. Rank the solutions in increasing order based on their aggregate costs.
7. Remove the solutions whose costs are larger than the cost of No Partition.
8. Based on the ranking choose the best solution.

Based on the chosen solution the vertical fragments are constructed. The next step involves partitioning each fragment further applying horizontal fragmentation. For each vertical fragment it is necessary to select the attributes and its corresponding predicates based on which the horizontal partitioning is performed. As the possible number of fragments produced is large, the fragmentation selection is carried out by using the proposed hybrid GAHC algorithm described in the previous step.

5.2.5 Fact Fragmentation

Using the proposed horizontal or mixed partitioning, the dimension table(s) are fragmented. The optimal fragment schema is chosen using the GAHC algorithm which is then partitioned by applying Range, List or Hash partitioning modes. Finally, based on this partition the fact table is referential partitioned in the underlying database.

The existing partitions need to be altered when the business analysis needs changes as new queries need to be imposed over the DW. Hence, the proposed approach provides a partition management component which monitors the DW query evolution and informs the DWA to perform the refragmentation as required.

5.3 PROPOSED PARTITION MANAGEMENT FOR EVOLVING QUERIES

In this section the proposed partition management for the ORP approach has been described. New queries arise due to changes in the business needs. Based on the frequency of the new queries, the existing partitions need to be adapted. This part of the proposed approach monitors the DW in order to collect statistics about queries, it detects any changes in query patterns and informs the DWA to trigger the refragmentation process. The Figure 5.5 represents the various steps involved in the partition management, which are explained in this section.

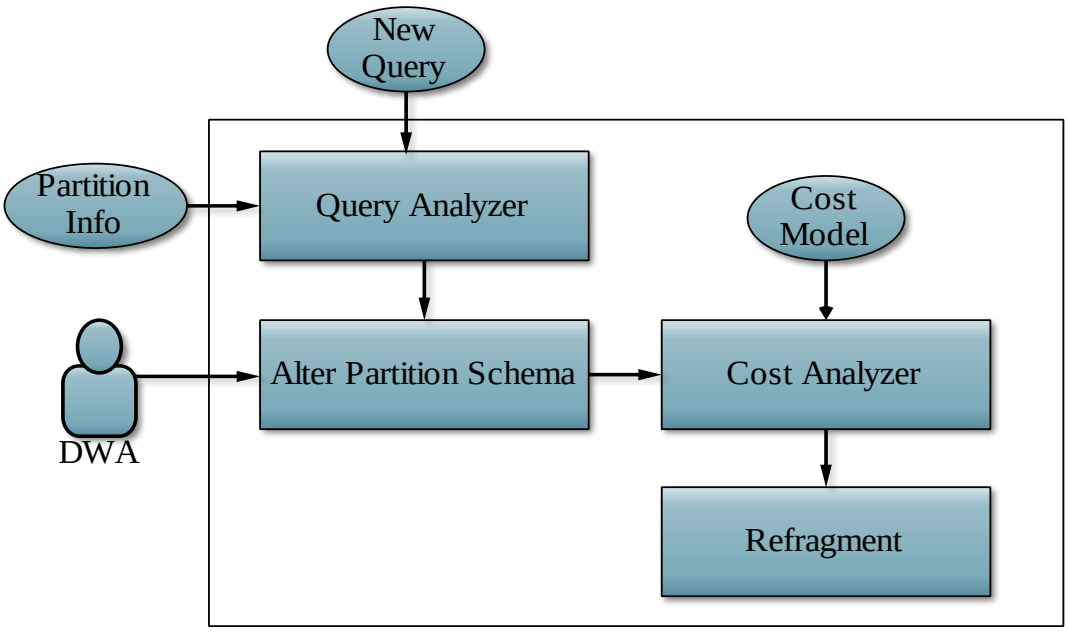

Figure 5.5 Partition Management

5.3.1 Query Analyzer

The first step involves analyzing a new query and inform the DWA for any alteration required in the existing partitions. In order to perform this task, information about the existing partitions, attributes of each partition and its values are maintained. The procedure given in Figure 5.6 is followed by the query analyzer. It works based on the following three cases to alert for the refragmentation process:

Case 1: When new values or predicates not in existing partitions exist in the query.
Case 2: When new attributes not in existing partitions exist in the query.
Case 3: When new table not in existing partitions exists in the query.

When new query q enters the existing query workload Q, then its frequency value FR is updated. As the same query appears repeatedly and when its frequency becomes greater than the threshold value, the query analyzer triggers the refragmentation (Steps 1-3). Here, the table names are first retrieved from the *select* clause of the query q (Step 4). When a table T_i in the query belongs to the partitioned table (PT) list, then the attributes of the partition are retrieved from the

where clause of the query. If this attribute $T_i.A_i$, is available in the partitioned attributes (PA) then its new value or its new predicate needs to be included in the existing partition. But if the attribute $T_i.A_i$ is not available in the partitioned attributes (PA) then refragmentation needs to be carried out for the table T_i with attribute A_i (Steps 5-12). When a table T_i in the query is not contained in a partitioned table (PT) list, then new fragmentation is triggered for this table T_i (Steps 13-16).

```
Trigger Refragment(Q, PT, PA)
1 Get new query q.
2 Update frequency of q; FR=FR+1;
3     if FR>FRThreshold then
4         for each Ti in select clause of q
5             if Ti □PT then
6                 for each Ti.Ai in where clause of Q
7                     if Ti.Ai □ PA then
8                         Refragment_NewValue (Ti, Ai,Value);
9                     else
10                        Refragment_NewAttribute (Ti, Ai,Value);
11                    end if
12                end for
13            else
14                Obtain Ti.Ai in where clause of Q
15                Fragmentation(Ti,Ti.Ai,Value);
16        end if
17    end for
18 end if
```

Figure 5.6 Trigger Refragmentation

5.3.2 Alter Partition Schema

When an alert is received for refragmentation, the designer or the DWA performs the modification of the schema of the received partition. The refragmentation may involve the addition of new fragmentation attribute or the alteration of an attribute's domain value or predicates. The two main operations that

are performed for modification over the partition schema are SPLIT and MERGE. The SPLIT operation divides the given partition to two separate partitions. And the MERGE is involved in merging two partitions into a single partition.

5.3.3 Cost Analyzer

This part of the management module calculates the cost of a partition schema before and after the alteration using the cost model given in equation 5.1. If the cost after alteration exceeds a certain cost threshold and the number of manageable fragments exceeds the fragmentation threshold, then the DWA can decide to retain the old partition. Otherwise, the altered schema of the partition is chosen for fragmentation.

5.3.4 Refragmentation

The final step executes the required alteration over the table partition in the underlying database. Thus, the proposed approach could adapt the existing partitions when the query imposed over the DW evolves.

In the proposed approach, the ORP along with the partition management provides an improved and automated (semi-automated) way of performing DW schema partitioning and its management in order to enhance the query performance. Following section discusses SSB (O'Neil *et al.,* 2007) case study in order to illustrate the steps involved in the ORP and to evaluate the proposed approach.

5.4 CASE STUDY – SSB (TPC-H)

The SSB benchmark is a variation of TPC-H benchmark. SSB is the DW schema for TPC-H domain. The SSB is a Sales DW schema which consists of the LineOrder, Customer, Date, Part and Supplier tables. The DW schema is given in Figure 5.7. The experimental setup and the evaluation of the proposed approach using the obtained results are discussed in section 5.5. Following are the steps of the

proposed ORP approach explained in section 5.2 and 5.3 which are applied to the case study:

1. Dimension Table and Attributes Selection
2. Fragment Schema Construction
3. Optimal Fragment Selection using GAHC
4. Fact Fragmentation
5. Partition Management
6. Optimized Mixed Fragmentation for Big Dimension

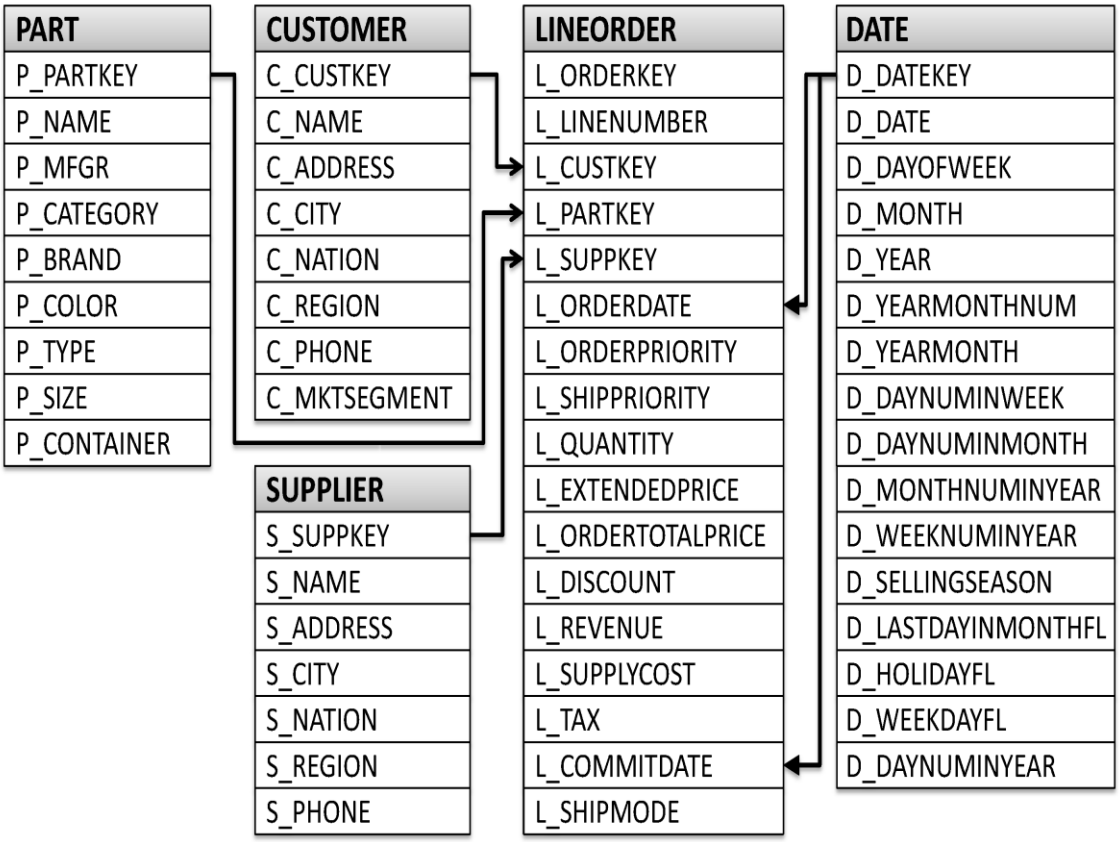

Figure 5.7 SSB Data Warehouse Schema

The SSB queries are used in the experiment for retrieving the results in the Sales DW and these queries are also used for computing the query execution time and query cost. The list of queries is given in Table 5.5.

Table 5.5 SSB Queries

Q.No.	Queries	Description
Q1.1	select sum(lo_extendedprice*lo_discount) as revenue from lineorder, date where lo_orderdate = d_datekey and d_year = 1998 and lo_discount between1 and 3 and lo_quantity < 25;	Quantifies the amount of revenue increase that results from eliminating certain companywide discounts in a given percentage range for products shipped in a given year.
Q1.2	select sum(lo_extendedprice*lo_discount) as revenue from lineorder, date where lo_orderdate = d_datekey and d_yearmonthnum = 199801 and lo_discount between4 and 6 and lo_quantity between 26 and 35;	
Q1.3	select sum(lo_extendedprice*lo_discount) as revenue from lineorder, date where lo_orderdate = d_datekey and d_weeknuminyear = 6 and d_year = 1998 and lo_discount between 5 and 7 and lo_quantity between 26 and 35;	
Q2.1	select sum(lo_revenue), p_brand1 from lineorder, date, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_category = 'MFGR#12' and s_region = 'AMERICA' and d_yearmonthnum = 199801group by p_brand1 order by p_brand1;	Compares revenue for some product classes, for suppliers in a certain region, grouped by more restrictive product classes and all years of orders.
Q2.2	select sum(lo_revenue), p_brand1 from lineorder, date, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand1 between 'MFGR#2221' and 'MFGR#2228' and s_region = 'ASIA' and d_yearmonthnum = 199801 group by p_brand1 order by p_brand1;	
Q2.3	select sum(lo_revenue), d_year, p_brand1 from lineorder, date, part, supplier where lo_orderdate = d_datekey and lo_partkey = p_partkey and lo_suppkey = s_suppkey and p_brand1 = 'MFGR#2221' and s_region = 'EUROPE' group by d_year, p_brand1 order by d_year, p_brand1;	
Q3.1	select c_nation, s_nation, d_year, sum(lo_revenue) as revenue from customer, lineorder, supplier, date where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_orderdate = d_datekey and c_region = 'ASIA' and s_region = 'ASIA' and d_year >= 1992 and d_year <= 1995 group by c_nation, s_nation, d_year order by d_year asc, revenue desc;	

Q3.2	select c_city, s_city, d_year, sum(lo_revenue) as reve-nue from customer, lineorder, supplier, date where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_orderdate = d_datekey and c_nation = 'UNITED STATES' and s_nation = 'UNITED STATES' and d_year >= 1992 and d_year <= 1995 group by c_city, s_city, d_year order by d_year asc, revenue desc;	Provides revenue volume for lineorder transactions by customer nation and supplier nation and year within a given region, in a certain time period.
Q3.3	select c_city, s_city, d_year, sum(lo_revenue) as reve-nue from customer, lineorder, supplier, date where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_orderdate = d_datekey and (c_city='UNITED KI1' or c_city='UNITED KI5');	
Q3.4	select c_city, s_city, d_year, sum(lo_revenue) as reve-nue from customer, lineorder, supplier, date where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_orderdate = d_datekey and (c_city='UNITED KI1' or c_city='UNITED KI5') and (s_city='UNITED KI1' or s_city='UNITED KI5') and d_yearmonth = 'Dec1997' group by c_city, s_city, d_year order by d_year asc, revenue desc;	
Q4.1	select d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit from date, customer, supplier, part, lineorder where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_partkey = p_partkey and lo_orderdate = d_datekey and c_region = 'AMERICA' and s_region = 'AMERICA' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, c_nation order by d_year, c_nation;	Measures the aggregate profit as (revenue – supplycost) group by year, nation and by product category.
Q4.2	select d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit from date, customer, supplier, part, lineorder where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_partkey = p_partkey and lo_orderdate = d_datekey and c_region = 'AMERICA' and s_region = 'AMERICA' and (d_year = 1997 or d_year = 1998) and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, s_nation, p_category order by d_year, s_nation, p_category;	
Q4.3	select d_year, s_city, p_brand1, sum(lo_revenue - lo_supplycost) as profit from date, customer, supplier, part, lineorder where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_partkey = p_partkey and lo_orderdate = d_datekey and c_region = 'AMERICA' and s_nation = 'UNITED STATES' and (d_year = 1997 or d_year = 1998) and p_category = 'MFGR#14' group by d_year, s_city, p_brand1 order by d_year, s_city, p_brand1;	

5.4.1 Dimension Table and Attributes Selection

The first step involves the selection of dimension table and attributes involved in the Sales schema. The DW schema ontology given in section 3.4.5 of Chapter 3 is considered for SSB Schema. This DWO and the SSB queries are given as input to DimSelection algorithm. For each dimension in the ontology the values of the parameters such as *frequency*, *attributes* and *size* are obtained. These values are given in Table 5.6. Based on the values, the scores for the dimension are included in the *dim_selection* matrix. The rows of the matrix are added and the total score for each dimension is shown in the Table 5.7.

Table 5.6 Parameter Values

Dimension	Frequency	Attributes	Size
Customer	7	4	30,000
Supplier	10	4	2000
Part	6	5	50,000
Date	13	4	2555

Table 5.7 DimSelection Matrix

Dimension	Frequency	Attributes	Size	Total Score
Customer	2	3	3	8
Supplier	3	3	1	7
Part	1	4	4	9
Date	4	3	2	9

The user can select one or more dimensions based on the scores. For example, if the Date dimension is selected for horizontal partitioning, its attributes need to be chosen. From the given set of SSB queries the *attribute_selection* matrix for the Date dimension is constructed as shown in Table 5.8. The 16 attributes of Date dimension are represented as A to P. According to the matrix, the d_datekey, d_year, d_yearmonthnum and d_weeknuminyear are the attributes available in the query. Based on the frequency the d_year and the d_yearmonthnum attributes are chosen for fragmentation.

Table 5.8 Attribute_Selection Matrix

	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	FREQ
A	1	1	1	1	1	1	1	1	1	1	1	1	1	PK
B														
C														
D														
E			1			1	1	1	1	1	1	1	1	9
F		1		1	1									3
G														
H														
I														
J														
K			1											1
L														
M														
N														
O														
P														

5.4.2 Fragment Schema Construction

The construction of fragment schema based on the chosen dimension table and its attributes is performed in this step. The domain values of the chosen attributes are obtained and based on these values the fragment schema is constructed as explained in section 5.2.2. The fragment schema holds integer values to represent the domain number. Table 5.10 represents the fragment schema for the Date dimension. The total possible fragments are 16 (4x4) for this dimension table as each attribute has 4 predicate values.

Table 5.9 Attribute Values for Date Dimension

Attribute Name	**Values**			
d_year	1992-1993	1994-1995	1996-1997	1998
d_yearmonthnum	199201-199312	199401-199512	199601-199712	199801-199812

Table 5.10 Fragment Schema for Date Dimension

Attribute Name	Domains			
d_year	1	2	3	4
d_yearmonthnum	1	2	3	4

5.4.3 Optimal Fragment Selection using GAHC

In order to apply the proposed GAHC algorithm to find the optimal number of fragments, the problem solution can be represented as given in Table 5.11. The following are the steps of the proposed algorithm:

i. **Initialization**: Ten initial solutions (fragmentation schemes) are produced randomly which is the population on which the algorithm works. A sample fragmentation schema is given in Table 5.11. The total number of generations is initialized 10, 50, 100 for each experiment. The threshold value is set, which is the maximum number of fragments N managed by the database.

Table 5.11 Initial Solution

Attribute Name	Domains			
d_year	1	2	3	4
d_yearmonthnum	1	2	3	4

ii. **Evaluation**: The fitness value for each and every fragmentation scheme is computed using the cost model given in equation 5.1.

iii. **Hill Climbing**: For each individual in the initial population hill climbing is applied. The number of fragments (N) generated by the individual is obtained. If N is greater than the threshold, then the sub domains in the fragment code (schema) of the individual are merged. For example the sub domains 1 and 2 of d_yearmonthnum can be merged as shown in Table 5.12. If N is lesser than the threshold, then the sub domains in the fragmentation code of the individual are splitted. After applying merge or split operations the fitness of the individual is calculated. When the individual has a higher

fitness than the original individual, it is retained for further improvements. After certain iterations when no further improvements are seen the hill climbing is terminated. The above steps are repeated for other individuals in the initial population.

Table 5.12. Initial Solution with Merging

Attribute Name	Domains			
d_year	1	2	3	4
d_yearmonthnum	1	2	3	4

iv. Genetic Operations:

a. Selection: Roulette wheel method is used in this algorithm. The two individuals with highest fitness value are chosen.

b. **Crossover:** New individual is created by crosses of the selected individuals. Here one-point crossover mechanism is used to give the same chances to the attributes with high and low number of sub domains. Figure 5.8 is an example for crossover operation.

c. **Mutation:** It involves in modifying the cells (genes) in the individual to obtain a new individual. Figure 5.9 is an example for mutation operation.

d. **New population:** The fitness value for each and every fragment is computed using the given cost model. These fragments form the new set of population.

v. **Termination:** If the termination condition is not satisfied, then the whole process is repeated. After a certain number of iterations the algorithm converges and the optimal fragmentation schema is obtained. The Table 5.13 represents the genetic algorithm parameters used in the proposed approach.

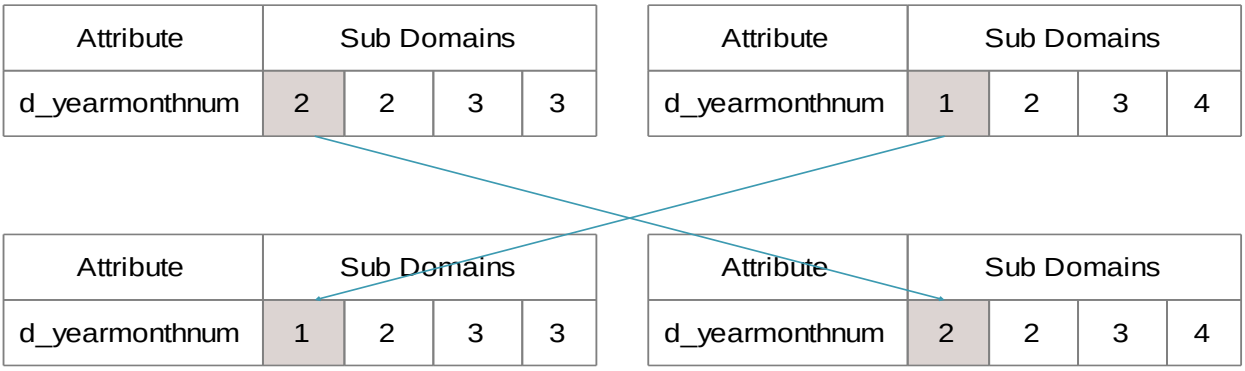

Figure 5.8 Cross over Operation

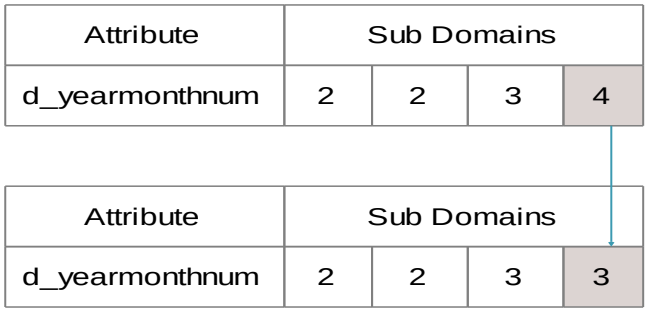

Figure 5.9 Mutation Operation

Table 5.13 Genetic Algorithm Parameters

Parameters	Values
Population Size	10
Maximum Generations	10, 50, 100
Encoding Mechanism	Decimal Encoding
Crossover	One point crossover
Selection	Roulette Wheel Method

5.4.4 Fact Fragmentation

After running the GAHC algorithm for fragmentation selection the optimal fragmentation schema is given in Table 5.14 that is used to partition the Sales DW. The corresponding domain values for the fragment schema are shown in Table 5.15.

Range partitioning can be applied for Date dimension table using d_year attribute and d_yearmonthnum attributes. Based on the partitions the LineOrder fact table can be referentially partitioned in the Oracle11G DBMS.

Table 5.14 Optimal Fragment Schema

Attribute Name	Domains			
d_year	1	1	3	4
d_yearmonthnum	1	1	1	4

Table 5.15 Optimal Fragment Schema with domain values

Attribute Name	Values			
d_year	1992-1995	1996-1997	1998	-
d_yearmonthnum	199201-199712	199801-199812	-	-

5.4.5 Partition Management

This part of ORP maintains information about existing partitions and performs the required alteration when new query arises. Table 5.16 provides the details of partition tables PT, and its corresponding partition attributes PA. For example, when a new query given in Figure 5.10 enters the query workload and its frequency is greater than the query frequency threshold FR, the existing partitions are verified by the query analyzer. According to the given query the partitions P0 and P1 are affected. Here, from the *where* clause of the query the d_year attribute has predicate as 1992-1997. Hence, the partitions P0 and P1 can be combined by performing the MERGE operation over these partitions.

Table 5.16 Partition Tables and Partition Attributes

Partition Table (PT)	PT0	PT1	PT2	PT3	PT4
Partition Attribute	d_year	d_year	d_year	d_yearmonthnum	d_yearmonthnum
Attribute Value	1992-1995	1996-1997	>1998	199201-199712	199801-199812

```
select c_nation, s_nation, d_year, sum(lo_revenue) as revenue from customer, lineorder, supplier,
date where lo_custkey = c_custkey and lo_suppkey = s_suppkey and lo_orderdate = d_datekey and
c_region = 'ASIA' and s_region = 'ASIA' and d_year >= 1992 and d_year <= 1997 group by
c_nation, s_nation, d_year order by d_year asc, revenue desc;
```

Figure 5.10 New Query

The resultant fragment schema after the modification is given in Table 5.17. The cost analyzer computes the cost of the new fragment schema using the cost model. As the computed cost is lesser than the cost threshold the partitions are merged at the physical level.

Table 5.17 Fragment Schema after merging

Attribute Name	Values	
d_year	1991-1997	1998

5.4.6 Optimized Mixed Fragmentation for Big Dimension

The SSB schema does not contain big dimension. Hence, the star schema of the Inmon's sales data mart (Inmon, 2005) has been used for applying the proposed partitioning technique (ORP-M) for big dimension. The data mart consists of a fact table and four dimension tables: Date Dimension, Product Dimension, Store Dimension and Customer Dimension as shown in Figure 5.11. Here the Customer dimension consists of 53 attributes which is chosen as the big dimension for partitioning. The DW has been populated using synthetic data set. A set of 8 OLAP queries are considered which are available for the star schema.

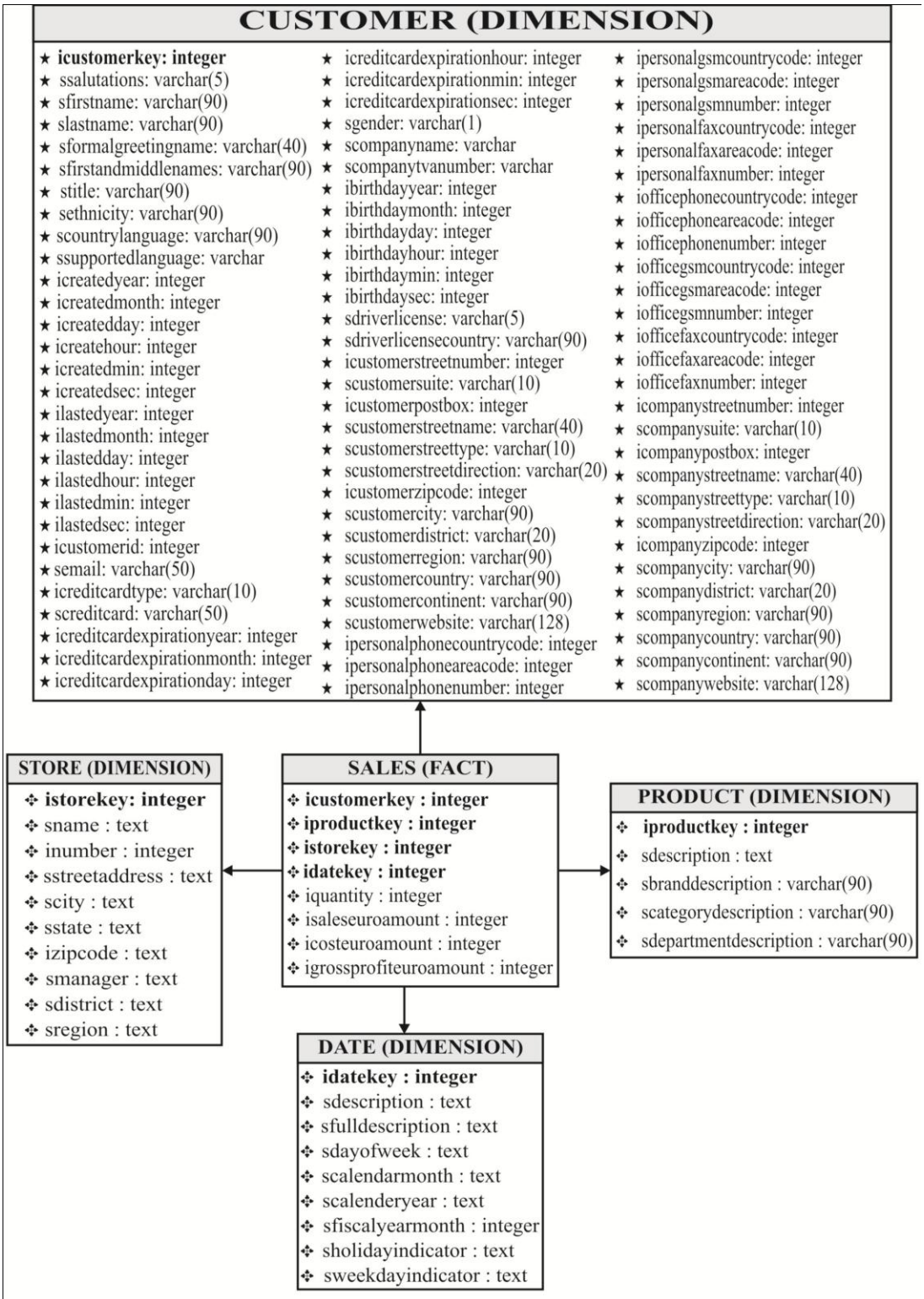

Figure 5.11 Inmon's Star Schema

Vertical partitioning explained in section 5.2.4 is performed over customer dimension by splitting the big dimension table into multiple tables, each of which contains different number of columns. To apply attribute clustering the QAM i.e query attributes matrix is constructed. The customer dimension attributes are given as (A1, A2, A3, A4, A5, A6, A7, A8, A9, A10….A53) = (CustomerId, FirstName, CountryLanguage, CustomerAge, DrivingLicense, PurchaseProduct, Purchaseyear, PurchaseQuantity, CustomerCountry, CustomerContinent, etc.)

Table 5.18 Sample QAM for Customer Dimension

Queries	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
Q1	1			1			1			
Q2	1		1							1
Q3	1		1					1	1	
Q4	1						1	1		
Q5	1	1		1						
Q6	1			1			1			
Q7	1						1	1		1
Q8	1			1	1					

Table 5.18 shows the QAM constructed for sample attributes of the customer dimension. The attribute sets formed using the QAM is given in Figure 5.12. After the attribute clustering is applied over the obtained sets using attribute clustering tree, the different possibilities of solutions for partitioning are given in Figure 5.13. For solutions S1 to S8 the query cost is estimated using the given cost model and the aggregate cost is shown in Table 5.19. From the table it is observed that S2 and S8 are the best solutions to vertically partition the customer table for the given set of sample attributes. The vertical partitions are represented in Table 5.20.

Attribute Set
{A1,A4,A7}
{A1,A3,A10}
{A1,A3,A8,A9}
{A1,A7,A8}
{A1,A2,A4}
{A1,A7,A8,A10}
{A1,A4,A5}

Figure 5.12 Attribute Sets

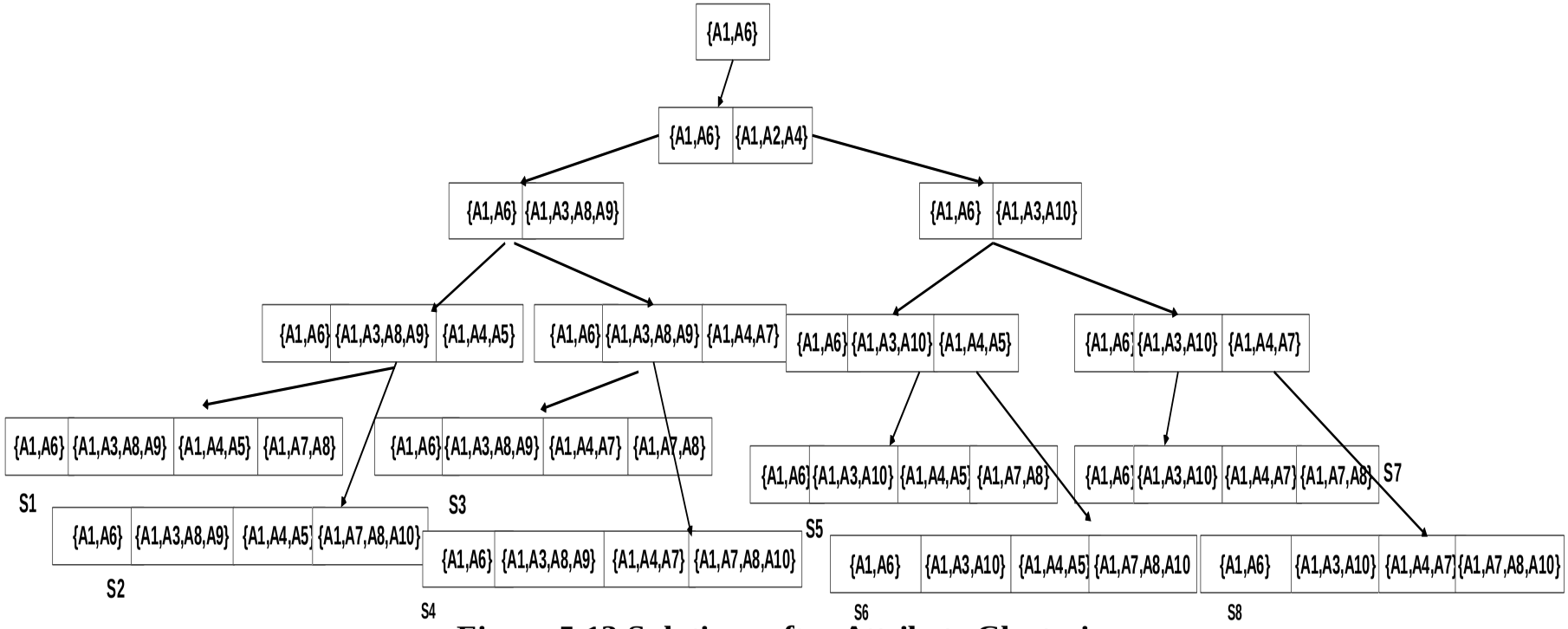

Figure 5.13 Solutions after Attribute Clustering

Table 5.19 Aggregate Cost for Partition Solutions

Solution	Estimated Query Cost								Aggregate Cost
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	
S1	9	49	98	9	49	44	90	9	44.63
S2	9	42	88	9	44	44	87	9	41.50
S3	9	47	98	9	49	47	90	9	44.75
S4	9	44	80	9	43	43	82	9	39.88
S5	22	49	84	20	49	44	90	22	47.50
S6	9	46	87	9	45	45	88	9	42.25
S7	24	49	98	22	43	43	98	29	50.75
S8	9	44	84	9	43	43	84	9	40.63

Table 5.20 Vertical Partitions

Vertical Partition 2	**Vertical Partition 2**	**Vertical Partition 3**	**Vertical Partition 4**
CustomerId	CustomerId	CustomerId	CustomerId
PurchaseProduct	CountryLanguage	CustomerAge	PurchaseYear
	PurchaseQuantity	PurchaseYear	PurchaseQuantity
	CustomerCountry		CustomerContinent

In order to horizontally fragment each vertical partition the attributes need to be selected along with the corresponding predicates to be used for fragmentation. For example, to fragment vertical partition 3 represented in Table 5.20, the fragmentation attributes are CustomerAge and PurchaseYear. The predicates of the attributes are obtained from the given specification which is given in Table 5.21. Based on the obtained predicates the fragmentation schema of the Vertical Fragment 3 (Customer) is shown in Table 5.22.

Table 5.21 Predicates for Horizontal Fragmentation

Attribute Name	**Values**			
Age	<38	38-48	48-58	>58
PurchaseYear	<2000	2000-2005	2005-2007	2007-2009

Table 5.22 Fragment Schema for Horizontal Fragmentation

Attribute Name	Domains			
Age	1	2	3	4
PurchaseYear	1	2	3	4

Based on the fragmentation schema given in Table 5.22 the total number of fragments is 16. The proposed GAHC algorithm is applied in order to find an optimal fragment schema which generates fragments, such that the overall query cost is minimized. As the fragment schema given in Table 5.23 gives the minimal query cost it is chosen as the optimal fragment.

Table 5.23 Optimal Schema for Horizontal Fragmentation

Attribute Name	Values			
Age	<38	38-48	48-58	>58
PurchaseYear	<2000	2000-2005	2005-2009	-

For partitioning the fact table, horizontal fragmentation has been applied to the Customer dimension using range partitioning mode for Age and PurchaseYear attributes. Based on these partitions the fact table Sales are referentially partitioned in Oracle11G DBMS.

5.5 RESULTS AND DISCUSSION

In this section, the experimental setup for evaluating the proposed approach is given. The results obtained are analyzed by providing a comparison of the proposed approach with the existing partitioning techniques. The comparative analysis is performed for the dimension selection methods, fragmentation selection algorithms and mixed fragmentation techniques.

5.5.1 Experimental Setup

An experimental study has been conducted to evaluate the proposed ORP strategy which performs referential partitioning for a given DW schema. The SSB schema has been used which is a Sales DW schema. The fact table LineOrder

contains 6,00,000 records, the dimension tables Customer has 30,000 records, Date has 2,555 records, Part has 50,000 records and Supplier has 2000 records. The experiments have been conducted using Oracle11G which supports referential partitioning. The dataset of SSB benchmark is created and populated using its data generator called DBGEN that enables the generation of synthetic data. This warehouse has been installed under Oracle 11g on a Pentium 1.8 GHz (with a memory of 256 mu, 60Go) running under windows XP pro.

5.5.2 Analysis of Dimension Selection Methods

The first experiment compares the quality of each parameter: *frequency*, *attributes* and *size* for dimension selection with the proposed matrix based selection. For each parameter the proposed referential partitioning is applied with different values of the threshold N. Where, N is the number of generated fragments to be managed in the underlying database. This N value is varied to 10, 20, 50 and 100 for each experiment. Here, it is assumed that the maximum threshold value is 100. The cost of evaluating the SSB queries is computed for the generated fragment schemas using the cost model given in equation 5.1. Table 5.24 summarizes the obtained results.

Table 5.24 Results for Dimension Selection Methods

Dimension Selection Methods	Fragments (10)	Fragments (10)	Fragments (10)	Fragments (10)
Frequency	17500000	14900000	17560000	17595000
Attributes	21700000	21750000	21700000	21800000
Size	21700000	21750000	21700000	21800000
Matrix_Based	17500000	14900000	17560000	17595000

From the Figure 5.14 it was observed that the matrix based selection produces minimal I/O cost for queries when compared to individual parameter for the dimension selection. In the given scenario, the Date dimension chosen by the matrix based selection is also given by the *frequency* parameter.

Hence, both results are same. Regarding the threshold values, it was observed that increasing the number of fragments does not reduce the query cost. Having the N value as 50 or 100 gives the same results. The reason is that, increasing the number of fragments might result in large number of join operations in the queries, which results in increased query cost.

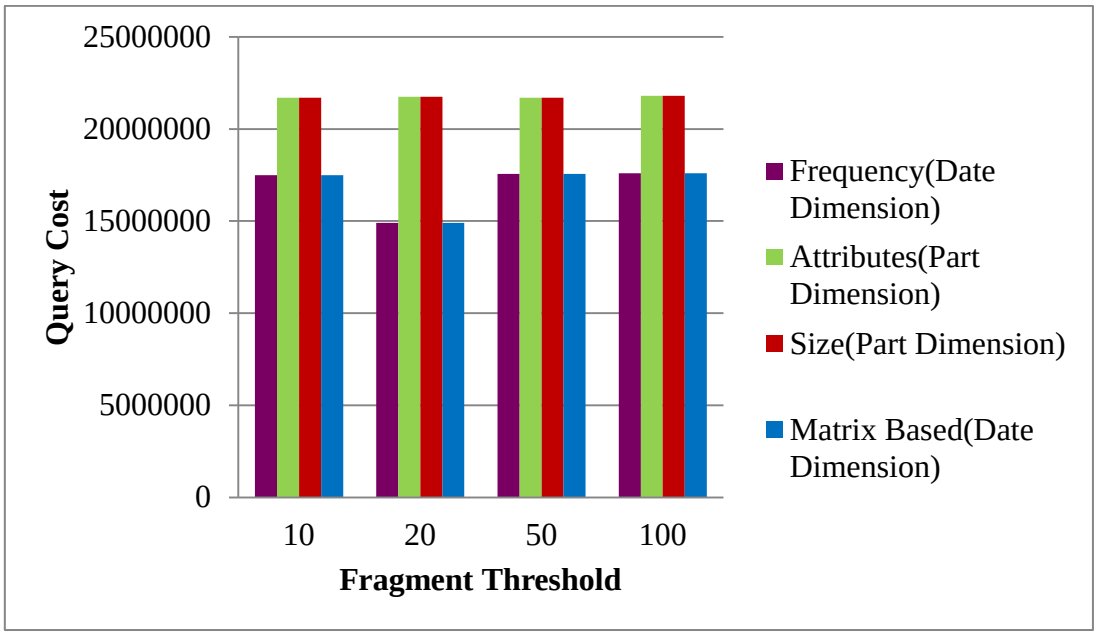

Figure 5.14 Comparison of Query Cost for Dimension Selection Methods

5.5.3 Analysis of Fragment Selection Methods

The SSB queries are executed over the Non_Partitioned DW and partitions generated by partitioned hill climbing (Partitioned_HC), partitioned genetic algorithm (Partitioned_GA), and the proposed GAHC algorithms. The results obtained for query execution time and query cost are given in Table 5.25.

Table 5.25 Results for Fragment Selection Methods

Queries	Query Execution Time (ms)				Query Cost			
	Non_Partitioned	Partitioned_HC	Partitioned_GA	Partitioned_GAHC	Non_Partitioned	Partitioned_HC	Partitioned_GA	Partitioned_GAHC
Q1.1	1789	1012	912	550	40800000	22000000	20000000	14500000
Q1.2	1845	1078	978	512	40100000	22050000	20010000	14900000
Q1.3	1678	1048	948	570	41500000	22700000	20005000	14560000
Q2.1	2189	1102	987	620	42900700	25800000	21008900	15595000
Q2.2	2235	1120	963	642	42700000	25000000	21000000	15500000
Q2.3	2412	1245	925	678	42800000	25050000	21310000	15900000
Q3.1	2190	1289	890	680	44990000	26700000	22005000	15560000
Q3.2	2213	1156	994	758	45990000	25800000	22008900	16595000
Q3.3	2460	1208	915	730	44790000	25000000	22006000	16540000
Q3.4	2372	1345	973	719	43790000	26050000	22005000	16530000
Q4.1	2864	1652	1082	806	45990000	26700000	22000000	17506000
Q4.2	2850	1521	1041	816	45890000	26700000	22004000	17507000
Q4.3	2812	1420	1090	860	47790000	25800000	22003000	17500000

Figure 5.15 provides the comparison for the individual query execution time when different fragment selection algorithms are applied. The proposed hybrid GAHC algorithm utilizing the advantages of hill climbing and genetic algorithm features selects the best optimal fragment schemas from the given search space. Hence the individual query execution time for the GAHC based partition is minimal for the given workload when compared to the partitions generated by existing fragment selection algorithms. The total query execution time when GAHC is applied is 8941 ms when compared to Non_Partitioned, Partitioned_HC and

Partitioned_GA which produced 299909 ms, 16196 ms and 12698 ms respectively as given in Figure 5.16.

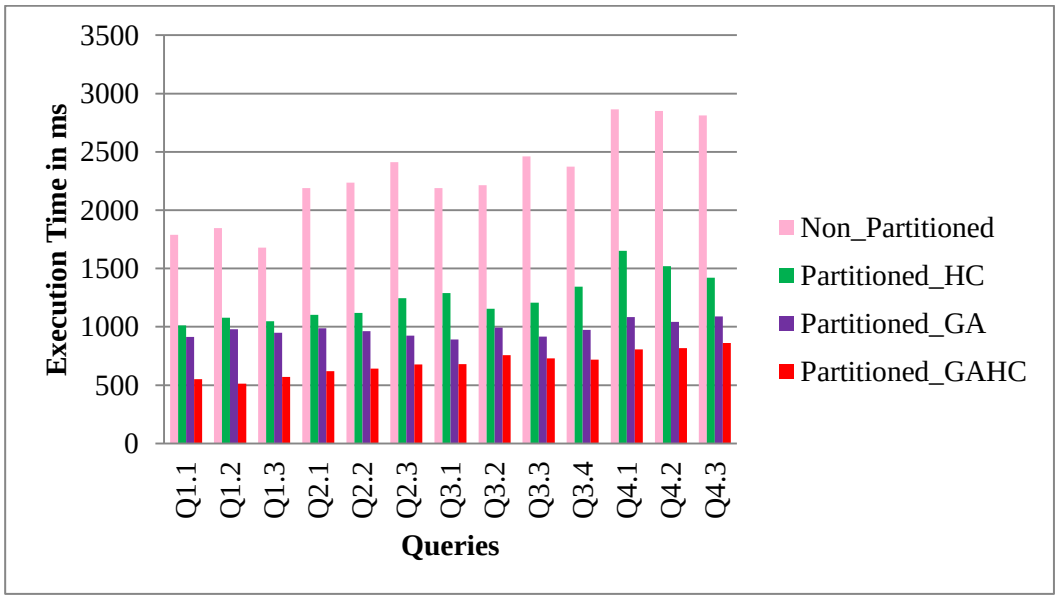

Figure 5.15 Comparison of Individual Query Execution Time for Fragment Selection Algorithms

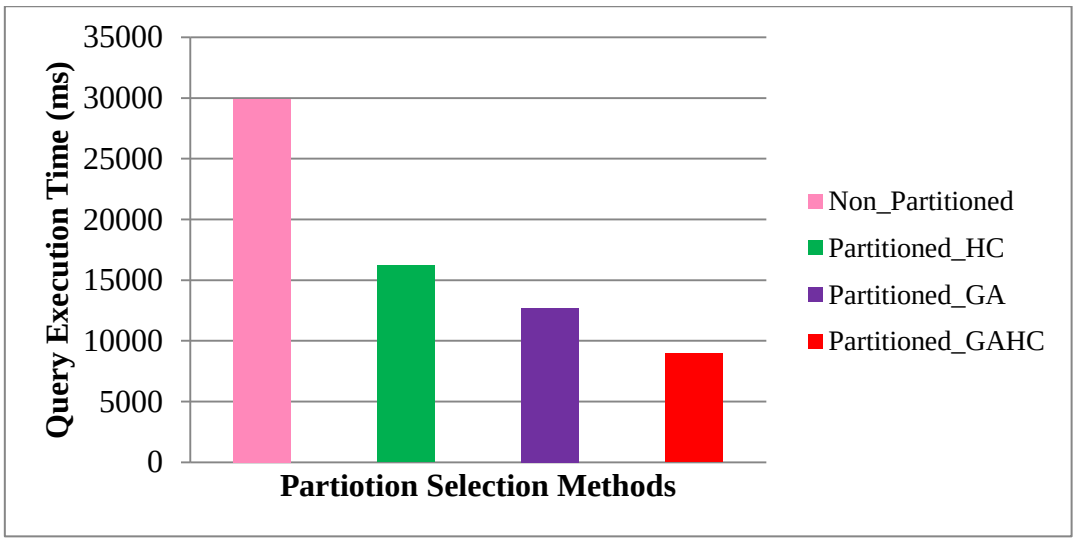

Figure 5.16 Comparison of Overall Query Execution Time for Fragment Selection Algorithms

In order to study the query cost for different partitioning techniques, the queries are executed over the fragment schemas generated and the cost is computed using the given cost model. From the Figure 5.17 it is observed that the fragment

schemas produced by GAHC gives minimal cost for individual queries. The reason is that, the number of I/Os required for executing the queries is minimal due to the optimal partition schema generated by the proposed GAHC partitioning technique. The overall cost of the given query is thus minimized when compared to Non_Partitioned, Partitioned_HC and Partitioned_GA as shown in Figure 5.18.

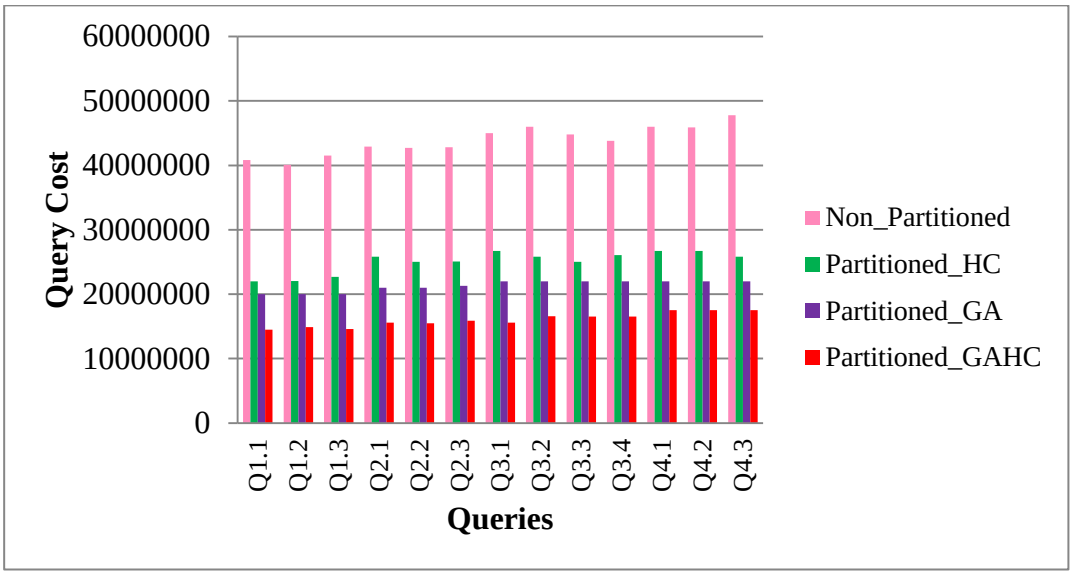

Figure 5.17 Comparison of Individual Query Cost for Fragment Selection Algorithms

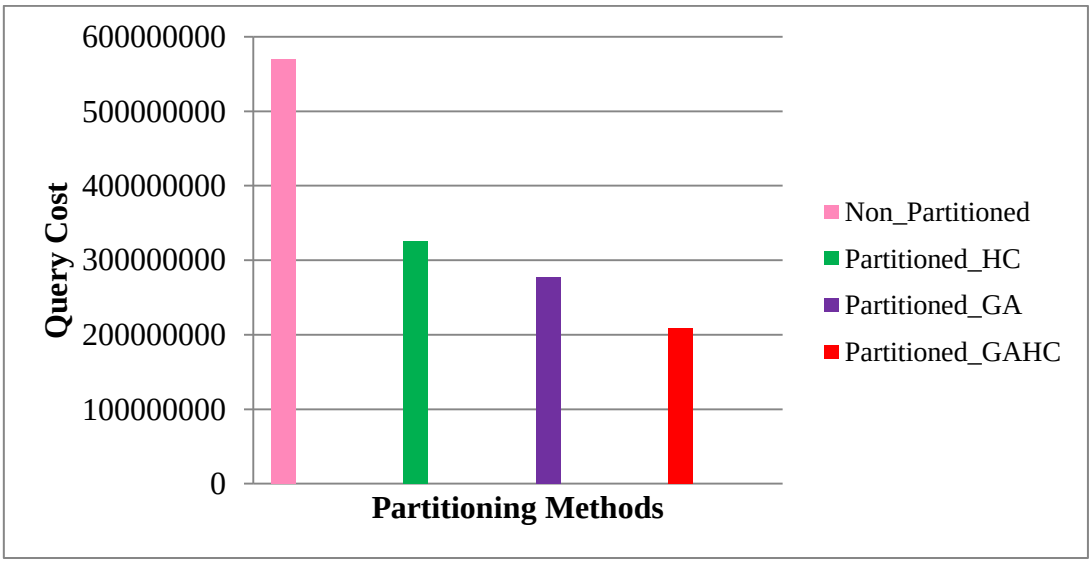

Figure 5.18 Comparison of Overall Query Cost for Fragment Selection Algorithms

5.5.4 Analysis of Mixed Fragmentation Techniques

The big dimension, i.e. customer dimension of Inmon's sales DW is partitioned using the traditional mixed fragmentation approach and the proposed ORP-M approach. By executing the queries over the partitioned dimension table and its corresponding fact table, the query execution time and query cost that are obtained are summarized in Table 5.26.

Table 5.26 Results for Mixed Fragmentation Techniques

Queries	Query Execution Time (ms)		Query Cost	
	Traditional Mixed Approach	ORP-M Approach	Traditional Mixed Approach	ORP-M Approach
Q1	4458	2058	2130000	1015300
Q2	8634	5632	5389000	3109000
Q3	3812	1914	1338900	1036500
Q4	9972	6970	5723800	4013800
Q5	3056	1055	2114000	1170000
Q6	7945	5846	5812000	2112200
Q7	4012	2017	3764100	1934700
Q8	1082	482	2753900	1445900

Figure 5.19 compares the individual query execution time (in ms) for the proposed ORP-M approach with the traditional mixed fragmentation. The proposed ORP-M uses optimized vertical partition based on attribute clustering followed by the horizontal partition of the vertical fragment. Hence, the partition generated by the ORP-M approach is optimal for executing the queries compared to the existing approach. Thus, from the figure 5.19 it is observed that ORP-M gives 25976 ms as

total query execution time, which is minimal compared to the traditional mixed approach which gives a total of 42597 ms.

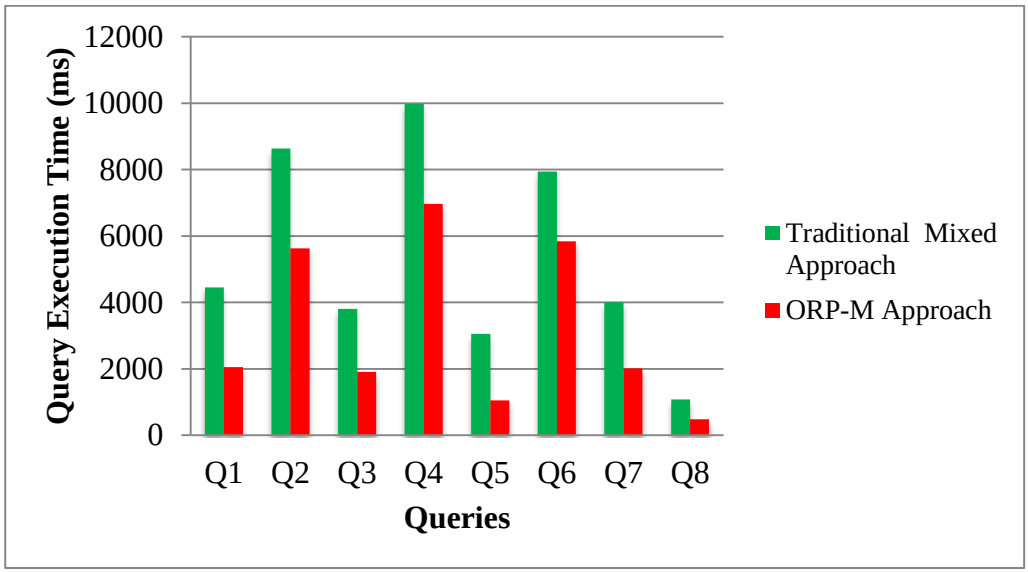

Figure 5.19 Comparison of Individual Query Execution Time for Mixed Approaches

The comparison of individual query cost for the given set of queries when applied to the mixed fragmentation approaches is given in Figure 5.20. The ORP-M performs better when compared to the traditional mixed approach.

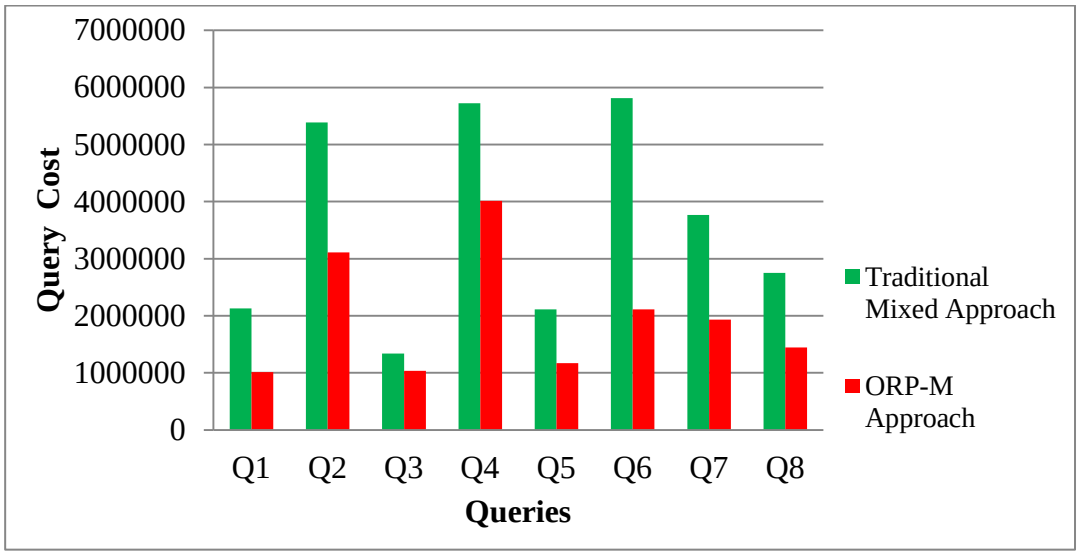

Figure 5.20 Comparison of Individual Query Cost for Mixed Approaches

From the obtained results it has been observed that the proposed ORP approach helps in reducing the query execution time and query cost when compared to existing partitioning techniques. Following inferences are made from the results show in Figure 5.15 to Figure 5.20:

i. The ORP-H approach optimized with matrix based dimension selection and hybrid GAHC for fragment selection performs better compared to existing approaches. Hence it could achieve 30% of reduced query execution time when compared to Partitioned_GA, 45% reduction compared to Partitioned_HC and 70% of reduction compared to Non_Partitioned approach.

ii. When comparing the query cost, ORP-H approach produces minimal cost. That is, 30% minimum cost compared to Partitioned_GA, 36% and 63% minimum compared to Partitioned_HC and Non_Partitioned approaches respectively.

iii. Comparing proposed ORP-M with the traditional mixed approach, the proposed approach provided 40% reduced query execution and 45% minimum query cost. The proposed mixed fragmentation could achieve better results as the vertical fragmentation is optimized with attribute clustering and horizontal fragmentation by adopting the proposed ORP-H approach.

5.6 SUMMARY

Partitioning plays an important role during the design of DW which helps to improve the performance of star join queries. As DW schema involves fact-dimension relationship referential partitioning gives greater benefits in terms of query performance when compared to single table partitioning. The selection of dimension and maintaining manageable numbers of fragments/partitions are the main issues of referential partitioning.

The proposed ORP approach provides a matrix based method, which uses multiple selection parameters for choosing the best dimension table(s) for partitioning. Based on the number of attributes and the values of the chosen dimension the number of possible fragments would be large. Hence the ORP approach provides hybrid GAHC (genetic and hill climbing) algorithm for optimal fragment selection based on the given cost model. The GAHC algorithm solves the limitations of the existing fragment selection algorithms and generates a valid set of fragments. With reference to the dimension fragments the fact table is partitioned horizontally.

The ORP-H is applied for horizontal fragmentation of the dimension table. Whereas, for partitioning a big dimension table ORP provides a mixed fragmentation (ORP-M) which involve optimized vertical partition using attribute clustering followed by the horizontal partition of the vertical fragment. The proposed ORP approach also provides facilities for refragmentation of existing partitions in case of evolving queries. It monitors the DW for changes in query pattern and informs the DWA for triggering the refragmetnation process.

The ORP is evaluated by applying it to a case study of SSB sales DW schema. The experiments are conducted for the existing and the proposed approach for DW schema partitioning. The obtained results are compared and it has been observed that the proposed approach provides better performance in terms of reducing query execution time and query cost.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENTS

The DW schema provides a multidimensional representation of the data integrated from several operational sources. As the DW involves a multifaceted environment the design and management of the multidimensional schema becomes a complex task. Hence, the overall objective of this research is to study the different issues that exist in this task and provide an improved solution for each. This chapter gives the conclusions of the research work with the research contributions and then outlines the possible future enhancement.

6.1 CONCLUSIONS

The first issue explored is the design of the multidimensional DW schema. The existing approaches tried to provide automation of the design task either from requirements or data source, but the results were not satisfactory. A new hybrid approach was developed utilizing both the requirements and data source knowledge to derive the multidimensional schema. It provides automation of the design task covering conceptual, logical and physical design phases by a formal representation of the requirements and data source concepts in ontology format. Experiments were conducted and the results show that the newly developed OntoMD approach outperforms the existing approaches in a significant way.

The second issue explored is the evolution of the DW schema for a changing business scenario. The existing methods on DW evolution handles schema changes at the physical level, hence it has an impact on the maintenance cost. Moreover, these methods failed to provide an automatic adaptation of dependent entities when the DW schema evolves. Thus, the cost of manual adaptation of the entities is high. A new method OntoEvol was developed which eliminates the deficiency of existing methods. The proposed method allows the propagation of changes from the requirements or the data source to the DW schema at the ontological level. And also, it provides an automatic adaptation of the dependent entities. The experiments

showed that the OntoEvol could effectively propagate the changes when compared to existing methods.

The final issue explored is to enhance the performance of referential partitioning of the DW schema. The existing works on partitioning focused only on fragment selection methods. They were not focused on the other problem exist in DW schema partitioning. Hence, a new optimized referential partitioning technique ORP was developed to provide improved solution for dimension table selection, attribute selection, fragment selection, big dimension fragmentation and query evolution. Comparisons were made with existing partitioning methods. The results show a significant improvement in query performance when ORP is applied for referential partitioning.

Following conclusion were derived based on the comparison of the proposed solutions with the existing approaches:

i. The newly proposed OntoMD approach could efficiently generate the multidimensional schema by the reconciliation of the knowledge contained in the data source and the requirements using ontology.
ii. In the OntoMD approach any ambiguity in the requirements is eliminated at the early stage of the conceptual design.
iii. It gives step by step guideline to generate the conceptual, logical and physical schema of the DW.
iv. An OntoMD tool was proposed in order to facilitate the designer to perform the design task in an automated way.
v. The DW schema quality generated by the OntoMD approach is better when compared the quality of schema produced by existing design approaches. OntoMD produce 13 to 15% improvement for correctness, 13 to 20% improvement for completeness, 10 to 19% improvement for minimality, 14 to 25% improvement for traceability and 13 to 22% improvement for interpretability metrics when compared to existing approaches.

vi. To handle evolution the newly proposed OntoEvol approach provides different evolution operators to propagate requirements as well as the source changes, automatically over the DW schema.

vii. The impact of a particular change is evaluated by the proposed approach before it is implemented at the physical level.

viii. The mapping between source and the DW, queries and views which are affected by a change are automatically adapted in the proposed OntoEvol approach.

ix. Compared to the existing evolution methods, the proposed OntoEvol approach has more effectiveness in propagating the changes over the DW schema. It produced 13% better results when compared to the existing MVTDW approach and 17% better than DWE approach.

x. The cost of automatic adaptation of dependent entities of the OntoEvol is minimal compared to the manual cost of adaptation. It could produce 62% minimal cost when compared to the manual adaptation.

xi. The ORP approach proposed optimizes the referential partitioning technique. In ORP-H, the best dimension(s) and its attributes are chosen for horizontal partitioning through matrix based selection.

xii. In the proposed ORP-M, the big dimension is partitioned, where the attribute clustering tree is used for choosing the vertical fragments on which the horizontal fragmentation is applied. This provides an optimized mixed fragmentation.

xiii. In ORP, a hybrid GAHC algorithm is developed to choose the optimal fragments and a refragmentation is applied in case of evolving queries.

xiv. The proposed ORP-H approach, thus minimizes the query execution time by 30% when compared to Partitioned_GA and 45% when compared to Partitioned_HC. It also minimizes the query cost by 36% when compared to Partitioned_GA and 63% when compared to Partitioned_HC.

xv. The proposed ORP-M approach minimizes query execution time by 40% and query cost by 45% when compared to the traditional mixed fragmentation technique.

Thus, in this research the developed approaches and techniques for the DW schema design and its management could solve several issues and provide promising results compared to the existing works.

The main focus of this research is to provide a systematic platform for any organization to handle the expensive and time consuming task of DW schema design and its management. Taking advantage of the ontology the DW designer can resolve the ambiguity present in the data source as well as the requirements and build the unified multidimensional schema. Hence, this research can provide an organization with a DW structure, that derives huge business benefits by providing accurate analysis of the past results, find correlations in the data available, and present information in a user-friendly way to business users.

6.2 FUTURE RESEARCH DIRECTIONS

There are several possibilities for the improvement of this research work. The OntoMD approach which generates simple dimension hierarchies could be further extended to form multiple dimension hierarchies. In case of DW evolution, the history of changes over the schema needs to be maintained as different versions. Hence, the OntoEvol can be further extended to support version management.

Further, the OntoEvol and ORP approaches can be integrated into the OntoMD tool such that the designer or administrator of the DW is given with a single point of access to design and management of the schema. As the design of ETL has received attention in recent research works, an integration of the DW schema and ETL design process can be carried out by ontology and other semantic web tools. Another interesting line of research is to design a data mart from the DW using ontology, where a data mart represents the subset of the DW and focused on specific function of the enterprise.

REFERENCES

1. Abelló, A., & Romero, O. (2010). Automatic validation of requirements to support multidimensional design. *Data & Knowledge Engineering*, Vol. 69 (9), pp. 917-942.
2. Banerjee, S., & Davis, K. C. (2009). Modeling data warehouse schema evolution over extended hierarchy semantics. In *Journal on Data Semantics XIII*, Springer, Berlin Heidelberg, pp. 72-96.
3. Barr, M. (2013) Bi-Objective Optimization Based on Compromise Method for Horizontal Fragmentation in Relational Data Warehouses. *International Journal of Machine Learning and Computing (IJMLC)*, Vol.3(3), pp. 250-254.
4. Bębel, B., Królikowski, Z., & Wrembel, R. (2006). Formal approach to modelling a multiversion data warehouse. *Bulletin of the Polish Academy of Sciences. Technical Sciences*, Vol. 54 (1), pp. 51-62.
5. Bellahsene, Z. (2002). Schema evolution in data warehouses. Knowledge and Information Systems, Vol. 4(3), pp. 283-304.
6. Bellatreche, L. (2012). Dimension Table Selection Strategies to Referential Partition a Fact Table of Relational Data Warehouses. *Recent Trends in Information Reuse and Integration*, pp. 19-41.
7. Bellatreche, L., & Woameno, K. Y. (2009). Dimension table driven approach to referential partition relational data warehouses. In *Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP*, ACM, pp. 9-16.
8. Bellatreche, L., Karlapalem, K., Mohania, M., & Schneider, M. (2000). What can Partitioning do for Your Data Warehouses and Data Marts?. In *Database Engineering and Applications Symposium, International*, IEEE Computer Society, pp. 437-437.
9. Benitez-Guerrero, E., Collet, C., & Adiba, M. (2004). THE WHES APPROACH TO DATA WAREHOUSE EVOLUTION. *e-Gn osis*. Vol. 2Art.

10. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, Vol. 284 (5), pp. 28-37.
11. Berson, A., & Smith, S. J. (1997). Data warehousing, data mining, and OLAP. McGraw-Hill, Inc..
12. Body, M., Miquel, M., Bédard, Y., & Tchounikine, A. (2002). A multidimensional and multiversion structure for OLAP applications. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, ACM, pp. 1-6.
13. Bog, A., Sachs, K., & Zeier, A. (2011). Benchmarking database design for mixed OLTP and OLAP workloads. In *Proceedings of the 2nd ACM/SPEC International Conference on Performance engineering*, ACM, pp. 417-418.
14. Boukhalfa, K., Bellatreche, L., & Alimazighi, Z. (2009). HP&BJI: A combined selection of data partitioning and join. In *New Trends in Data Warehousing and Data Analysis*, Springer US, pp. 1-23.
15. Brkić, L., Baranović, M., & Mekterović, I. (2012). Improving the completeness and timeliness by horizontal fragmentation of data warehouse tables. In *Proceedings of the 11th international conference on Telecommunications and Informatics, Proceedings of the 11th international conference on Signal Processing*, World Scientific and Engineering Academy and Society (WSEAS), pp. 76-81.
16. Buitelaar, P., Olejnik, D., & Sintek, M. (2004). A protégé plug-in for ontology extraction from text based on linguistic analysis. In The Semantic Web: Research and Applications, Springer Berlin Heidelberg, pp. 31-44.
17. Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., & Wilkinson, K. (2004, May). Jena: implementing the semantic web recommendations. *In Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM, pp. 74-83.
18. Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. *ACM Sigmod record*, Vol. 26 (1), pp. 65-74.
19. Chen, S., Zhang, X., & Rundensteiner, E. A. (2006). A Compensation-Based Approach for View Maintenance in Distributed Environments. *IEEE*

Transactions on Knowledge and Data Engineering, Vol. 18 (8), pp. 1068-1081.

20. Cho, V., & Ngai, E. W. (2003). Data mining for selection of insurance sales agents. Expert systems, Vol. 20(3), pp. 123-132.
21. Conn, S. S. (2005). OLTP and OLAP data integration: a review of feasible implementation methods and architectures for real time data analysis. In *SoutheastCon, 2005. Proceedings. IEEE*, pp. 515-520.
22. Council, T. P. P. (2005). Transaction processing performance council. *Web Site, http://www. tpc. org*.
23. Council, T. P. P. (2008), TPC-H benchmark specification [Online]. Available: www.tpc.org/tpch/
24. Cruz, I. F., & Xiao, H. (2005). The role of ontologies in data integration. *International journal of engineering, intelligent systems for electrical engineering and communications*, Vol. 13 (4), pp. 245-252.
25. Cullot, N., Ghawi, R., & Yétongnon, K. (2007). DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In *15th Italian Symposium on Advanced Database Systems (SEBD)*, pp. 491-494.
26. Curino, C., Moon, H. J., & Zaniolo, C. (2009). Automating database schema evolution in information system upgrades. In *Proceedings of the 2nd International Workshop on Hot Topics in Software Upgrades*, ACM, pp. 1-5.
27. Dimovski, A., Velinov, G., & Sahpaski, D. (2011). Horizontal partitioning by predicate abstraction and its application to data warehouse design. In *Advances in Databases and Information Systems*, Springer Berlin Heidelberg, pp. 164-175.
28. Eadon, G., Chong, E. I., Shankar, S., Raghavan, A., Srinivasan, J., & Das, S. (2008). Supporting table partitioning by reference in oracle. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, ACM, pp. 1111-1122.
29. Eder, J., Koncilia, C., & Kogler, H. (2002). Temporal data warehousing: business cases and solutions. In *4th International Conference on Enterprise Information Systems* (*ICEIS)*, pp. 81-88.

30. Eder, J., Koncilia, C., & Wiggisser, K. (2006). Maintaining temporal warehouse models. In *Research and Practical Issues of Enterprise Information Systems*, Springer US, pp. 21-30.
31. Fan, H., & Poulovassilis, A. (2004). Schema evolution in data warehousing environments–a schema transformation-based approach. In *Conceptual Modeling–ER 2004*, Springer Berlin Heidelberg, pp. 639-653.
32. Frias, L., Queralt, A., & Ramon, A. O. (2003). EU-Rent car rentals specification.
33. Gagnon, M. (2007). Ontology-based integration of data sources. In*Information Fusion, 2007 10th International Conference on* IEEE, pp. 1-8.
34. Gardner, S.(1998). Building the data warehouse. *Comm. ACM*, Vol. 41 (9), pp. 52–60.
35. Giorgini, P., Rizzi, S., & Garzetti, M. (2008). GRAnD: A goal-oriented approach to requirement analysis in data warehouses. *Decision Support Systems*, Vol. 45 (1), pp. 4-21.
36. Glorio, O., Pardillo, J., Mazón, J. N., & Trujillo, J. (2008). Dawara: An eclipse plugin for using i* on data warehouse requirement analysis. In *International Requirements Engineering (RE'08). 16th IEEE*, pp. 317-318.
37. Golfarelli, M., & Rizzi, S. (1998). A methodological framework for data warehouse design. In *Proceedings of the 1st ACM international workshop on Data warehousing and OLAP*, ACM, pp. 3-9.
38. Golfarelli, M., & Rizzi, S. (1999). Designing the data warehouse: Key steps and crucial issues. Journal of Computer Science and Information Management, Vol. 2(3), pp. 88-100.
39. Golfarelli, M., & Rizzi, S. (2009). *Data warehouse design: Modern principles and methodologies*. McGraw-Hill, Inc..
40. Golfarelli, M., Lechtenbörger, J., Rizzi, S., & Vossen, G. (2006). Schema versioning in data warehouses: Enabling cross-version querying via schema augmentation. *Data & Knowledge Engineering*, Vol. 59 (2), pp. 435-459.
41. Golfarelli, M., Maio, D., & Rizzi, S. (1998). The dimensional fact model: a conceptual model for data warehouses. *International Journal of Cooperative Information Systems*, Vol. 7 (02n03), pp. 215-247.

42. Gomez-Perez, A., Fernández-López, M., & Corcho-Garcia, O. (2004). Ontological engineering. *Computing Reviews*, Vol. 45 (8), pp. 478-479.
43. Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, Vol. 5 (2), pp. 199-220.
44. Guo, Y., Pan, Z., & Heflin, J. (2005). LUBM: A benchmark for OWL knowledge base systems. W*eb Semantics: Science, Services and Agents on the World Wide Web*, Vol. 3(2), pp.158-182.
45. Howe, D. C. (2009). Rita wordnet. Java based API to access Wordnet. [Online]. Available: http://www.rednoise.org/rita/wordnet/documentation/
46. Hüsemann, B., Lechtenbörger, J., & Vossen, G. (2000). Conceptual data warehouse design. In *Proc. of the International Workshop on Design and Management of Data Warehouses (DMDW)*, pp. 3–9.
47. Inmon, W. H. (2005). *Building the data warehouse*. John wiley & sons.
48. Jovanovic, P., Romero, O., Simitsis, A., & Abelló, A. (2012). ORE: an iterative approach to the design and evolution of multi-dimensional schemas. In *Proceedings of the fifteenth international workshop on Data warehousing and OLAP*, ACM, pp. 1-8.
49. Kalnis, P., & Papadias, D. (2001). Proxy-server architectures for OLAP. In *ACM SIGMOD Record*, ACM, Vol. 30 (2), pp. 367-378).
50. Kimball, R. (Ed.). (1998). The data warehouse lifecycle toolkit: expert methods for designing, developing, and deploying data warehouses. John Wiley & Sons.
51. Kimball, R., & Ross, M. (1996). The Data Warehouse Toolkit., John Wiley&Sons. *Inc., New York*.
52. Kimball, R., & Ross, M. (2002). The data warehouse toolkit: the complete guide to dimensional modelling. *US: John Wiley & Sons*.
53. Lacoste, D., Sawant, K. P., & Roy, S. (2011). An efficient XML to OWL converter. In *Proceedings of the 4th India software engineering conference*, ACM, pp. 145-154.
54. Lechtenbörger, J., & Vossen, G. (2003). Multidimensional normal forms for data warehouse design. *Information Systems*, Vol. 28 (5), pp. 415-434.

55. Liu, X., & Iftikhar, N. (2013). Ontology-Based Big Dimension Modeling in Data Warehouse Schema Design. In *Business Information Systems*, Springer Berlin Heidelberg, pp. 75-87.

56. Luján-Mora, S., & Trujillo, J. (2003). A Comprehensive Method for Data Warehouse Design. In *Proc. of the 5th Intl. Workshop on Design and Management of Data Warehouses (DMDW'03)*, Berlin, Germany, pp. 1.1–1.14.

57. Mahboubi, H., & Darmont, J. (2009). Enhancing xml data warehouse query performance by fragmentation. In *Proceedings of the 2009 ACM symposium on Applied Computing*, ACM, pp. 1555-1562.

58. Malinowski, E., & Zimányi, E. (2006). Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering*, Vol. 59(2), pp. 348-377.

59. March, S. T., & Hevner, A. R. (2007). Integrated decision support systems: A data warehousing perspective. *Decision Support Systems*, Vol. 43(3), pp. 1031-1043.

60. Mazón, J. N., & Trujillo, J. (2009). A hybrid model driven development framework for the multidimensional modeling of data warehouses!. *ACM SIGMOD Record*, 38(2), pp. 12-17.

61. McGuinness, D. L., & Van Harmelen, F. (2004). OWL web ontology language overview. *W3C recommendation*, 10 (10), 2004.

62. Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

63. Moody, D. L., & Kortink, M. A. (2000, June). From enterprise models to dimensional models: a methodology for data warehouse and data mart design. In *Proceedings of the Second Intl. Workshop on Design and Management of Data Warehouses (DMDW*), p. 5.

64. Niedrite, L., Solodovnikova, D., Treimanis, M., & Niedritis, A. (2007). Goal-Driven Design of a data warehouse based business process analysis system. In *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases*, pp. 243-249.

65. Noy, N. F., & Musen, M. A. (2003). The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, Vol 59 (6), pp. 983-1024.

66. O'Neil, P., O'Neil, E. J., & Chen, X. (2007). The star schema benchmark [Online]. Available: http://www.cs.umb.edu/~poneil/StarSchemaB.PDF (Accessed 12 November 2013).

67. Ontology, P. (2007). Knowledge Acquisition System [Online]. Available: http://protege. stanford. edu.

68. Oueslati, W., & Akaichi, J. (2011). A multiversion trajectory data warehouse to handle structure changes. *International Journal of Database Theory and Application*, Vol. 4 (2), pp. 35-49.

69. Ozsu, M. T., & Valduriez, P. (1999) Principles of Distributed Database Systems, Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

70. Papadomanolakis S. and Ailamaki A. (2004) Autopart: Automating schema design for large scientific databases using data partitioning. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004)*, pp. 383–392.

71. Papastefanatos, G., Vassiliadis, P., Simitsis, A., & Vassiliou, Y. (2007). What-if analysis for data warehouse evolution. In *Data Warehousing and Knowledge Discovery*, Springer Berlin Heidelberg, pp. 23-33.

72. Pardillo, J., & Mazón, J. N. (2011). Using ontologies for the design of data warehouses. *International Journal of Database Management Systems*, Vol. 3 (2), pp. 73-87.

73. Peralta, V. and Ruggia, R., 2003. Using Design Guidelines to improve Data Warehouse logical Design [Online]. Available: ttp://www.fing.edu.uy/inco/grupos/csi/esp/Cursos/cur

74. Pérez, J. M., Berlanga, R., Aramburu, M. J., & Pedersen, T. B. (2008). Integrating data warehouses with web data: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 20 (7), pp. 940-955.

75. Poe, V., Brobst, S., & Klauer, P. (1997). Building a data warehouse for decision support. Prentice-Hall, Inc..

76. Prat, N., Akoka, J., & Comyn-Wattiau, I. (2006). A UML-based data warehouse design method. *Decision Support Systems*, Vol. 42(3), pp. 1449-1473.

77. Rechy-Ramirez, E. J., & Benitez-Guerrero, E. (2006). A model and language for bitemporal schema versioning in Data Warehouses. In *Computing, 2006. CIC'06. 15th International Conference on* IEEE, pp. 309-314.

78. Romero, O., & Abelló, A. (2010). A framework for multidimensional design of data warehouses from ontologies. *Data & Knowledge Engineering*, Vol. 69 (11), pp. 1138-1157.

79. Romero, O., Simitsis, A., & Abelló, A. (2011). GEM: requirement-driven generation of ETL and multidimensional conceptual designs. In *Data Warehousing and Knowledge Discovery*, Springer Berlin Heidelberg, pp. 80-95.

80. Sahpaski, D., Velinov, G., Jakimovski, B., & Kon-Popovska, M. (2009). Dynamic evolution and improvement of data warehouse design. In *Informatics, 2009. BCI'09. Fourth Balkan Conference in* IEEE, pp. 107-112.

81. Sanjay A., Narasayya V. R., and Yang B. (2004) Integrating vertical and horizontal partitioning into automated physical database design. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 359–370.

82. Selma, K., Ilyès, B., Ladjel, B., Eric, S., Stéphane, J., & Michael, B. (2012). Ontology-based structured web data warehouses for sustainable interoperability: requirement modeling, design methodology and tool. *Computers in Industry*. Vol. 63(8), pp-799–812.

83. Selman, B., and Gomes, C. (2006). Hill-climbing search. In *Encyclopedia of Cognitive Science*, John Wiley & Sons.

Skoutas, D., Simitsis, A., & Sellis, T. (2009). Ontology-driven conceptual design of ETL processes using graph transformations. *In Journal on Data Semantics XIII*, Springer Berlin Heidelberg, pp. 120-146.

84. Smith, M. K., Welty, C., & McGuinness, D. L. (2004). OWL Web Ontology Language Guide. W3C [Online]. Available: http://www.w3.org/TR/owl-guide/.

85. Solodovnikova, D., & Niedrite, L. (2011). Evolution-oriented user-centric data warehouse. In *Information Systems Development*, Springer, New York, pp. 721-734.

86. Song, I. Y., Khare, R., & Dai, B. (2007). SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, ACM, pp. 9-16.

87. Srivastava, J., & Chen, P. Y. (1999). Warehouse creation-a potential roadblock to data warehousing. *IEEE Transactions on Knowledge and Data* Engineering, Vol. 11(1), pp. 118-126.

88. Thakur, G., & Gosain, A. (2011). DWEVOLVE: a requirement based framework for data warehouse evolution. *ACM SIGSOFT Software Engineering Notes*, Vol. 36 (6), pp. 1-8.

89. Van Harmelen, F., and McGuinness, D. (2003). OWL Web Ontology Language Overview [Online]. Available: http://www.w3.org/TR/2003/WD-owl-features-20030331.

90. Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on IEEE*, pp. 18-22.

91. Winter, R., & Strauch, B. (2004). Information requirements engineering for data warehouse systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, ACM, pp. 1359-1365.

92. Xuan, D. N., Bellatreche, L., & Pierra, G. (2006). A versioning management model for ontology-based data warehouses. In *Proceedings of the 8th international conference on Data Warehousing and Knowledge Discovery* , Springer-Verlag, pp. 195-206.

LIST OF PUBLICATIONS

International Journals

1. M. Thenmozhi, K. Vivekanandan, “An Ontology based Hybrid Approach to Derive Multidimensional Schema for Data warehouse”, *International Journal of Computer Application*, Vol. 54, No.8, ISSN (Online): (0975 – 8887) pp. 36-42, September 2012.
2. M. Thenmozhi, K. Vivekanandan, “A Tool for Data Warehouse Multidimensional Schema Design using Ontology”, *International Journal of Computer Science Issues*, Vol. 10, Issue 2, No. 3, ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784, pp.161–168, March 2013.
3. M. Thenmozhi , K. Balachandar, and K. Vivekanandan. "An Ontology Mapping Maintenance Approach Using Change History Log in Ontology Based Data Integration." *CiiT International Journal of Data Mining and Knowledge Engineering*, Vol. 5, No. 7, ISSN(Online): 0974 – 9578,ISSN(Print):0974-9683, pp. 292-300, 2013.
4. M. Thenmozhi, and K. Vivekanandan. "A Combined Algorithm for Data Warehouse Fragmentation Selection. “ *International Journal of Engineering Science and Technology*, Vol. 6 No.5, ISSN : 0975-5462, pp. 240-253, May 2014.
5. M. Thenmozhi and K. Vivekanandan, “An Ontological Approach to Handle Multidimensional Schema Evolution for Data Warehouse”, *International Journal of Database Management Systems*, Vol. 6, No.3, ISSN : 0975 – 5985, pp. 33-52, June 2014.
6. M. Thenmozhi, K. Vivekanandan. " A Mixed Fragmentation Approach for Solving Big Dimension Problem in Data Warehouse ",*International Journal of Engineering Research & Technology*, Vol. 3 - Issue 6, ISSN: 2278-0181, pp. 1984-1900, June - 2014.
7. M. Thenmozhi and K. Vivekanandan,” Data Warehouse Schema Evolution and Adaptation Framework Using Ontology ” *International Journal on*

Computer Science and Engineering, Vol. 6 No.07, ISSN: 0975-3397, pp. 232-246, Jul 2014.

8. M. Thenmozhi and K. Vivekanandan," A Hybrid Multidimensional Modeling Approach for Data Warehouse Using Ontology*" Inderscience, International Journal of Business Information Systems*. (accepted)

International Conferences

1. M. Thenmozhi, K. Vivekanandan, "A Semi-automatic Approach to Update Mapping for Ontology Evolution", *Proceedings of Second International Conference on Computational Intelligence and Information Technology, CIIT 2012, **LNICST***, ISSN: 1867-8211, pp. 131–136, December 2012.
2. Thenmozhi M, Vivekanandan K and Gayathree G, "A-Semi Automatic Approach to Handle Data Warehouse Schema Evolution using Ontology" *Proceedings of Fourth International Joint Conference – AET 2013, NCR (**Elsevier**),* pp. 55-65, December 2013, India.
3. M. Thenmozhi and K. Vivekanandan," A Combined Heuristic Approach for Horizontal Partitioning of Data Warehouse" ***IEEE** International Conference on Advances in Engineering and Technology - 2014*. ISBN No.: 978-1-4799-4949-6 *(accepted)*
4. M. Thenmozhi and K. Vivekanandan," A Comparative Analysis of Fragmentation Selection Algorithms for Data Warehouse Partitioning", ***IEEE** International Conference on Advances in Engineering &Technology Research* ISBN No.: 978-1-4799-6393-5/14 *(accepted)*