AN ONTOLOGICAL APPROACH TO HANDLE MULTIDIMENSIONAL SCHEMA EVOLUTION FOR DATA WAREHOUSE

M.Thenmozhi and K.Vivekanandan

Department of Computer Science and Engineering,
Pondicherry Engineering College, Puducherry, India

ABSTRACT

In recent years, the number of digital information storage and retrieval systems has increased immensely. Data warehousing has been found to be an extremely useful technology for integrating such heterogeneous and autonomous information sources. Data within the data warehouse is modelled in the form of a star or snowflake schema which facilitates business analysis in a multidimensional perspective. As user requirements are interesting measures of business processes, the data warehouse schema is derived from the information sources and business requirements. Due to the changing business scenario, the information sources not only change their data, but also change their schema structure. In addition to the source changes the business requirements for data warehouse may also change. Both these changes results in data warehouse schema evolution. These changes can be handled either by just updating it in the DW model, or can be developed as a new version of the DW structure. Existing approaches either deal with source changes or requirements changes in a manual way and changes to the data warehouse schema is carried out at the physical level. This may induce high maintenance costs and complex OLAP server administration. As ontology seems to be a promising solution for the data warehouse research, in this paper an ontological approach to automate the evolution of a data warehouse schema is proposed. This method assists the data warehouse designer in handling evolution at the ontological level based on which decision can be made to carry out the changes at the physical level. We evaluate the proposed ontological approach with the existing method of manual adaptation of data warehouse schema.

KEYWORDS

Data Warehouse Evolution, Multidimensional Schema Evolution, Data Warehouse Schema Changes

1. INTRODUCTION

Business analysts increasingly rely on data warehouse for making strategic decisions for their organization. The data warehouse is populated from several operational sources using extraction, transformation and loading (ETL) operations [8]. In order to discover trends and critical factors in business, the data warehouse provides multidimensional view of the data which is used by Online Analytical Processing (OLAP) and other reporting tools [3]. A retail organization for example, analyzes purchases by its customers to come up with products of likely interest to the customer. The analytical results generated by these tools need to be accurate and reliable. Hence

the management of such data warehouse system is rather difficult when compared to traditional operational systems.

The common assumption is that once created the DW remains static. However, due to changes in business needs, the DW needs to evolve. Evolution in a DW can be generated by two different causes: (i) a change in the source schema or (ii) a change in the DW requirements [19]. An inherent feature of source schema is their evolution in time with respect not only to their contents (data) but also to their structures (schemas). In addition to this, the business processes in which the analyst is involved are subject to frequent changes. These changes in business processes are reflected in the analysis requirements. New types of queries that require different data become necessary. The new query requirements lead to changes in the DW schema [9]. These changes need to be propagated to the data warehouse system in order to provide accurate in-sight of the business data.

Considering the importance of the evolution problem in the data warehouse domain, some existing works have addressed this issue. In the literature, the DW evolution can be classified into three different approaches namely schema evolution [1] [2] and schema versioning [14] [15] [17]. Schema evolution consists in replacing old schema into a new schema as well transferring old data from old schema and updating it in a new schema. Schema versioning consists in keeping the history of all versions by temporal extension or by physical storing of different versions.

Existing works to the data warehouse evolution problems mainly concentrated on handling the DW schema changes at the physical level. As a consequence, the DW administrator is responsible for complex maintenance task either for schema evolution approach or versioning approach. The proposed work gives a different perspective for data warehouse evolution. It helps the data warehouse designer to give an insight of how the requirement or source changes can be propagated to the data warehouse schema using ontology. Hence, the designer can make a choice of propagating the changes to the physical level.

2. LITERATURE SURVEY

In [15] the authors proposed an approach based on versioning called MVTDW (Multiversion Trajectory Data Warehouse) in order to handle structural changes of the DW. They defined a set of constraints that need to be guaranteed by the real versions and alternative versions maintained by the MVTDW. Moreover they proposed some algorithms that can be applied in case of schema and instance changes on the TDW versions. For MVTDW to answer the queries the user needs to define the correct version. If data required by a query exists in different versions, it is necessary provide a data mapping between them. In this paper [17] the authors proposed the metadata model for a multi-version data warehouse. The metadata is used to describe the data warehouse schema versions at logical level, their storage in the relational database at physical level, information about reports defined by users on schema versions, and semantics of data stored in the data warehouse. The proposed data warehouse evolution framework enables the user to construct report based on desirable terms and term versions from the semantic metadata. In this paper [1] the authors formally defined the data warehouse model supporting extended hierarchies. Different features such as multiple hierarchies, non-covering hierarchies, non-onto hierarchies, and non-strict hierarchies are explained. They use Uni-level Description Language (ULD) and multilevel dictionary definition (MDD) in order to model the constructs. In this paper [4] the

authors developed the multi-version data warehouse (MVDW) approach to manage DW evolution. Since in MVDW fact and dimension data are physically distributed among multiple DW versions, the authors proposed a multi-version join index (MVJI) technique. The MVJI has an upper level for indexing attributes and a lower level for indexing DW versions. The paper also presents the theoretical upper bound analysis of the MVJI performance characteristic with respect to I/O operations. In [18] the authors proposed a framework for DW evolution based on requirements. The framework consists of Requirement level; which keep tracks of changes in requirements, Change Management level; keeps track of changes in source, Design level; to perform the required changes in the data warehouse schema, and View level; to define and generate customized reports and views. In this paper [2] the authors proposed a Rule-based Data Warehouse (R-DW) model to support data warehouse evolution. They follow a user-driven approach in order to update the data warehouse schema. Based on users' knowledge the aggregated data is represented in the form of "if-then" rules. Using these aggregation rules the granularity of the dimension are defined dynamically.

3. BACKGROUND

The following sections describe about the multidimensional model for data warehouse and the basics of ontology and its applicability in data warehouse domain.

3.1. Multidimensional Model

Data within the data warehouse is represented as Multidimensional model. This "cube" structure representation makes more compatible logical data representation suitable for On-Line Analytical Processing (OLAP) and data management. The advantages of dimensional modelling are: (i) To produce database structures that are easy for end-users to understand and write queries against, and (ii) To maximize the efficiency of queries. The basic concepts of dimensional modelling are: facts, dimensions and measures [10]. A fact is a collection of related data items, consisting of measures and context data. In general it represents business items or business transactions of a particular domain. A dimension is a collection of data that describe one business dimension. It determines the contextual background for the facts and they are the parameters over which we want to perform OLAP operations. A measure represents the numeric attribute of a fact which provides the performance or behaviour of the business relative to the dimensions. There are two basic models that are used in dimensional modelling: Star model and Snowflake model [3]. The basic structure for a dimensional model is the star model. It has one large central table (fact table) and a set of smaller tables (dimensions) arranged in a radial pattern around the central table as shown in Figure 1. The Figure 1 represents the star schema the sales domain. The snowflake model is the result of decomposing one or more of the dimensions. There exist many-to-one relationships among sets of attributes of a dimension which can separate new dimension tables, forming a hierarchy as shown in Figure 2. The decomposed snowflake structure visualizes the hierarchical structure of dimensions very well.

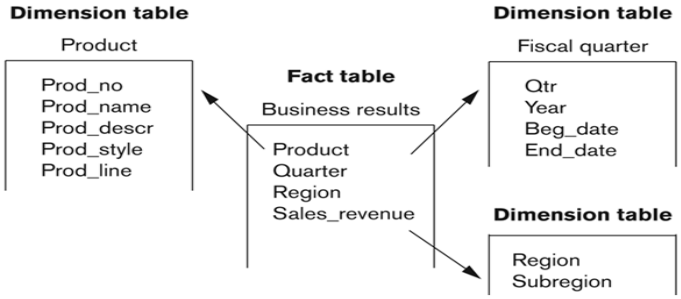

Figure 1. Star Schema

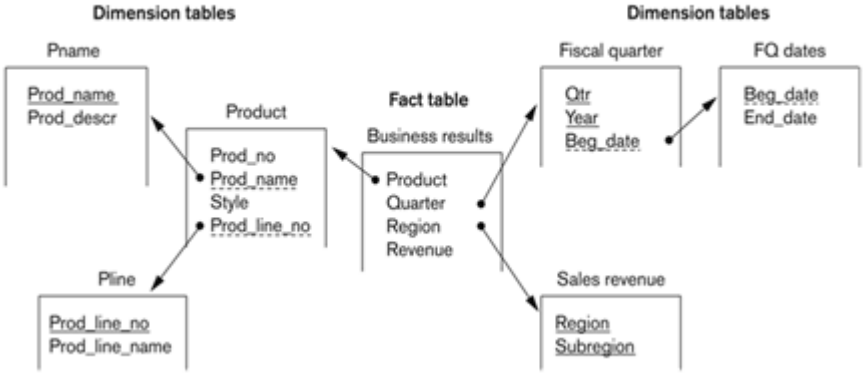

Figure 2. Snow Flake Schema

3.2. Ontology Basics

A relational database is a structured, formal representation of a set of data; in the same way, ontology is a structured, formal representation of an area of knowledge. It defines and restricts what can be said about that area of knowledge. Ontology is most commonly defined as "a formal, explicit specification of a shared conceptualization" [7]. Ontology describes the knowledge in a domain in terms of classes and properties, as well as relationships between them"[7]. Basic building blocks of ontology design include: classes or concepts, properties of each concept describing various features and attributes of the concept, restrictions on slots (facets). Classes (Concepts) are abstract groups, sets, or collections of objects. Concepts in the ontology should be close to objects (physical or logical) and relationships in the domain of interest. The decision in using an ontology-based approach for data warehouse, instead of using another technology for example a UML-based approach, lies in the fact that ontology provides an elegant way to represent concepts using Web Ontology Language (OWL) format. The OWL is an international standard for encoding and exchanging ontologies [16]. 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.

4. PROPOSED APPROACH

The objective of the work described in this paper is to propose a methodology that supports an automatic adaptation of the multidimensional schema when the requirements and source evolves. Our main idea for the data warehouse evolution is that the whole system is specified at the conceptual level. Any changes at the source or requirements are propagated to the data warehouse schema at the ontological level. Hence the proposed approach analyses the impact of a given change at the logical level, before it is propagated to the data warehouse schema at the physical level.

In our work we represent the data source, requirements and data warehouse using ontology. The ontological representation of these entities helps us to automate the evolution task. As the data source schema changes, the data warehouse schema need to evolve. Any changes at the original source or requirements are obtained and these are updated at the source and requirement ontology. Depending on the type of change, the changes are propagated over the data warehouse ontology. Based on the evolved data warehouse ontology the DW administrator can make a decision to carry the changes at the physical level. Following are the phases of our methodology:

1. Formalizations of inputs
2. Defining Evolution operators
3. Update change in requirements or source ontology
4. Extract Data Warehouse schema change
5. Propagate change to Data Warehouse schema
6. Check consistency of Data Warehouse schema

4.1. Formalization of Inputs

In this section, we formalize the proposed method in order to standardize and to ensure the correctness of the DW evolution process. We use OWL ontology to describe the semantics of different entities involved in our methodology. The reason for using OWL as the utility, instead of XML, UML or others, is that OWL supports the semantic reasoning, and is better for future extensions of this work, such as, reasoning-based DW schema evolution. We describe the data source, requirements and DW schema by ontology.

We illustrate our approach using TPC-H [5] and Star Schema Benchmark (SSB) [12] schemas. TPC-H is a decision support benchmark which represents our source. The Star Schema Benchmark is a variation of the TPC-H benchmark, which models the data warehouse for the TPC-H schema. The following sections describe our approach in detail.

4.1.1. Ontology for data sources

Data sources may contain different types of structures. Hence these sources are mapped to a local ontology which will express the semantic of the data sources. Following are the rules to map a database to ontology:

- The database table is mapped to an ontology class.
- If a database table is related to another, then the two tables are mapped to classes with parents-child relationship.

- If a database table is related to two tables, then the table is divided into two transferred classes.
- The primary key is mapped to a data type property of the ontology. The foreign key would be mapped to an object property of the ontology.
- The attributes of a table are mapped to properties of the equivalent class.

In the case that source data is extract from distributed data sources, more than one initial ontologies could be generated. Ontology merging [11] is used to merge different initial ontology together to represent a single integrated source. Figure 3 represents the ontology for the TPC-H schema. Formally, the data source ontology (DSO) to be used by our approach can be defined:

DSO = {C,DP,OP}

- C is a set of OWL classes representing the tables
- DP is a set of data type properties representing the table attributes
- OP is a set of object type properties representing the relationship between tables

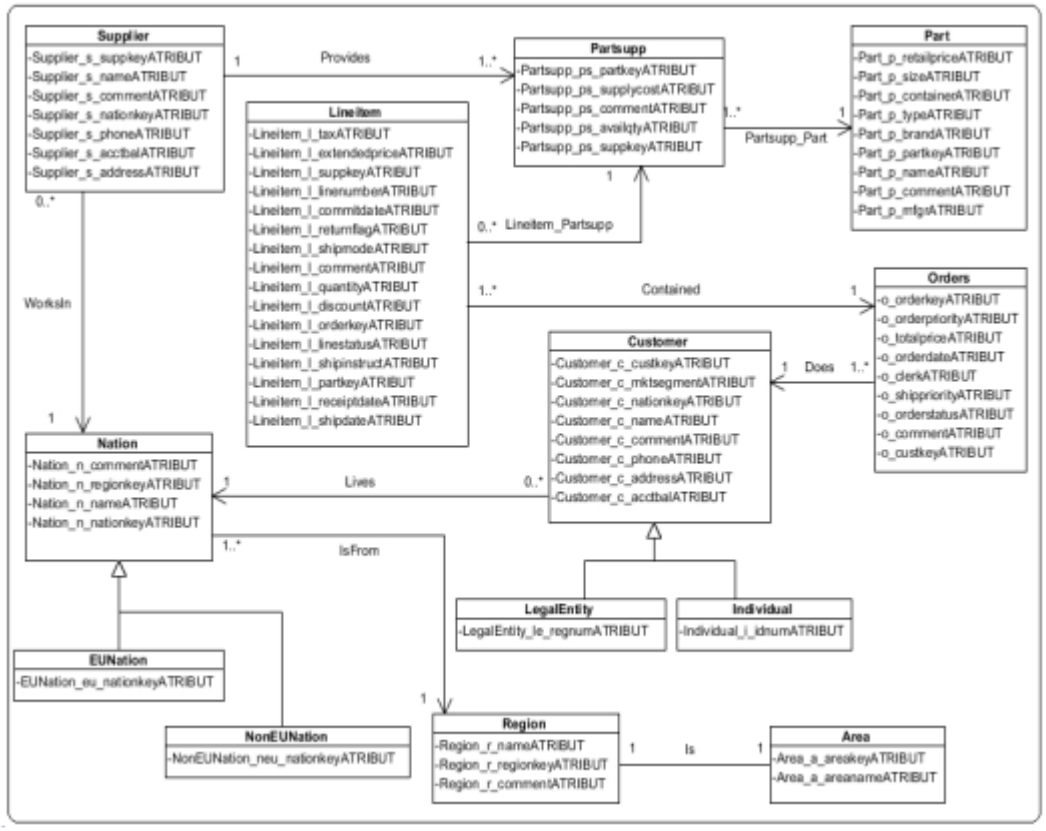

Figure 3. Data Source Ontology

4.1.2. Ontology for Requirements

Requirements play a major role in DW design. We assume that a requirement analysis has been carried out earlier and the corresponding requirement specification is available for the business domain. Here we explain the process for constructing the DW requirements ontology (DWRO)

for semantically describing the requirement glossaries. DWRO is based on the i* framework [6] for the modelling of goals and information requirements for DWs. The DWRO should be capable to model the following type of information: i) Strategic goals: These are the main objectives of the business process (for example, ‘‘increase sales”); ii) Decision goals: They attempt to answer the question: ‘‘how can a strategic goal be achieved?” (for example, ‘‘determine some kind of promotion”); iii) Information goals: They attempt to answer the question: ‘‘how can decision goals be achieved in terms of information required” (for example, ‘‘analyze customer purchases” or ‘‘examine stocks”). Information requirements for decision makers are derived from information goals. The multidimensional elements are the process measures under analysis (Measure stereotype), and context of analysis (Context stereotype). Figure 4 represents the requirement ontology based on TPC-H benchmark schema. Formally, the DWRO can be defined:

DWRO = {S, I, D, IR, BP,M,C}, 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 object properties representing business process
- M is a set of data type properties representing measures
- C is a set of OWL classes representing the contexts

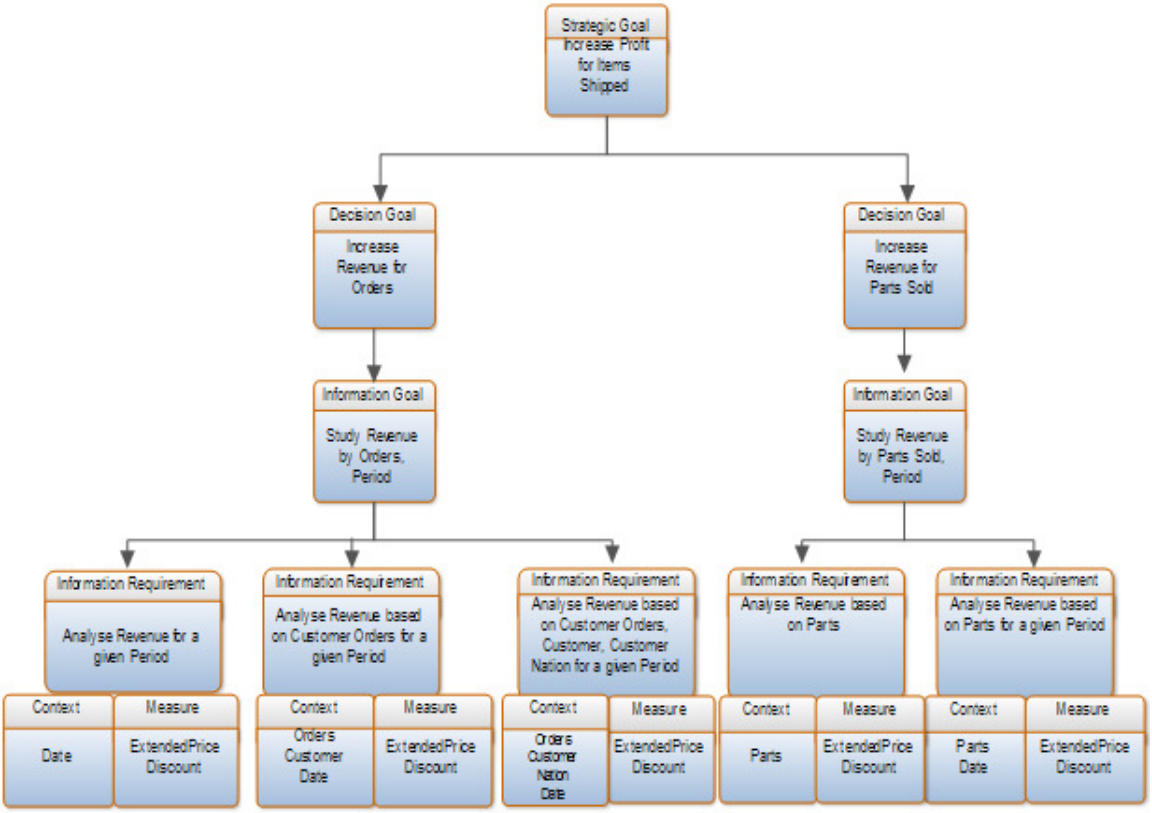

Figure 4. Data Warehouse Requirement Ontology

4.1.3. Ontology for Data Warehouse

The SSB data warehouse schema for the TPC-H source schema is given in Figure 5. DW schema can be formally defined using DWO as given below:

DWO = {F,FP,M,D,DP,RP}

- F is a set of OWL classes representing the fact
- FP is a set of OWL classes representing the fact properties
- M is a set of data type properties representing the measures of the fact
- D is a set of OWL classes representing the dimensions
- DP is a set of data type properties representing the dimension properties
- RP is a set of object properties representing the relationship between facts and dimensions

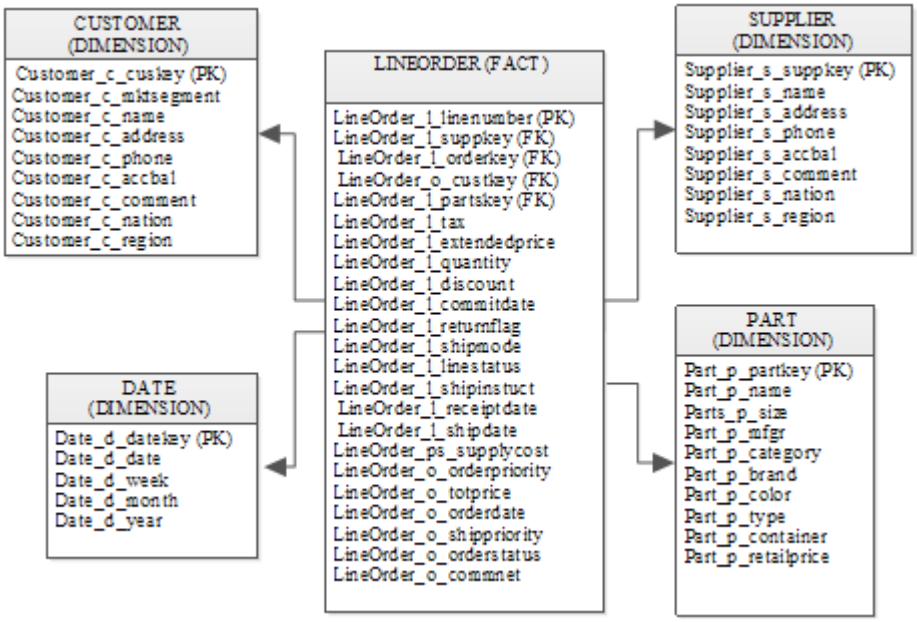

Figure 5. Data Warehouse Ontology

4.2. Defining Evolution Operators

In this section we present the set of evolution operators to represent the type of change and concept changed. The three possible changes that occur are addition, deletion and rename. The DW elements such as Fact, Dimension, Measures etc., are subject to change. Their equivalent ontology concepts in DWO need to be changed accordingly. Performing a change over the DWO may require additional changes to be executed over the ontology. For example, addition of a new dimension i.e., 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 1.

Table 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
	Attributes (Measures, 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
	Attributes (Measures, 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)
	Attributes (Measures, Descriptive)	Data Property	Rename Data Property (If required)
	Relationship (Primary Key, Foreign Key)	Object Property	Rename Object Property (If required)

4.3 Update Change in Requirements or Source Ontology

Given the changes in source schema or business requirements, the next step is to update them in the corresponding ontology. Using an ontology editor such as protégé [13] the changes can be directly applied over the ontology. When there is change in the business requirements, before the change is propagated over the DWO, the DSO need to be verified for the existence of the particular concept. Algorithm 2 *SearchOntology* verifies the existence of the concept. Steps 1-6 verifies for class existence, steps 7-12 verifies for data property existence, steps 8-17 verifies for object property existence. If concept not found the business requirements need to be refined.

```
SearchOntology (Ontology, Concept, Concept_Type)
1       if Concept_Type ==Class then
2               for all c_i ∈ C do
3                       if c_i == Concept then
4                               Flag=Found
5                       end if
6               end for
7       else if Concept_Type ==DataProperty then
8               for all dp_i ∈ DP do
9                       if dp_i == Concept then
10                              Flag=Found
11                      end if
12              end for
13      else if Concept_Type ==ObjectProperty then
14              for all op_i ∈ OP do
15                      if op_i == Concept then
16                              Flag=Found
17                      end if
18              end for
19      else
20              Flag=NotFound
21      end if
22      return Flag
```

Figure 6. Algorithm Search Ontology

For our TPC-H domain a new decision goal “Increase Revenue through promotions” has been added in the requirement ontology. “Study Revenue by Customer Promotions” is the information goal and the corresponding information requirement is “Analyse 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”. Figure 7 highlights the new requirement added to the requirement ontology.

Table 2 represents the details changes of a new table/class “Promotions” and its corresponding attributes/data properties that have been added to the TPC-H source schema ontology and other changes such as rename and deletions.

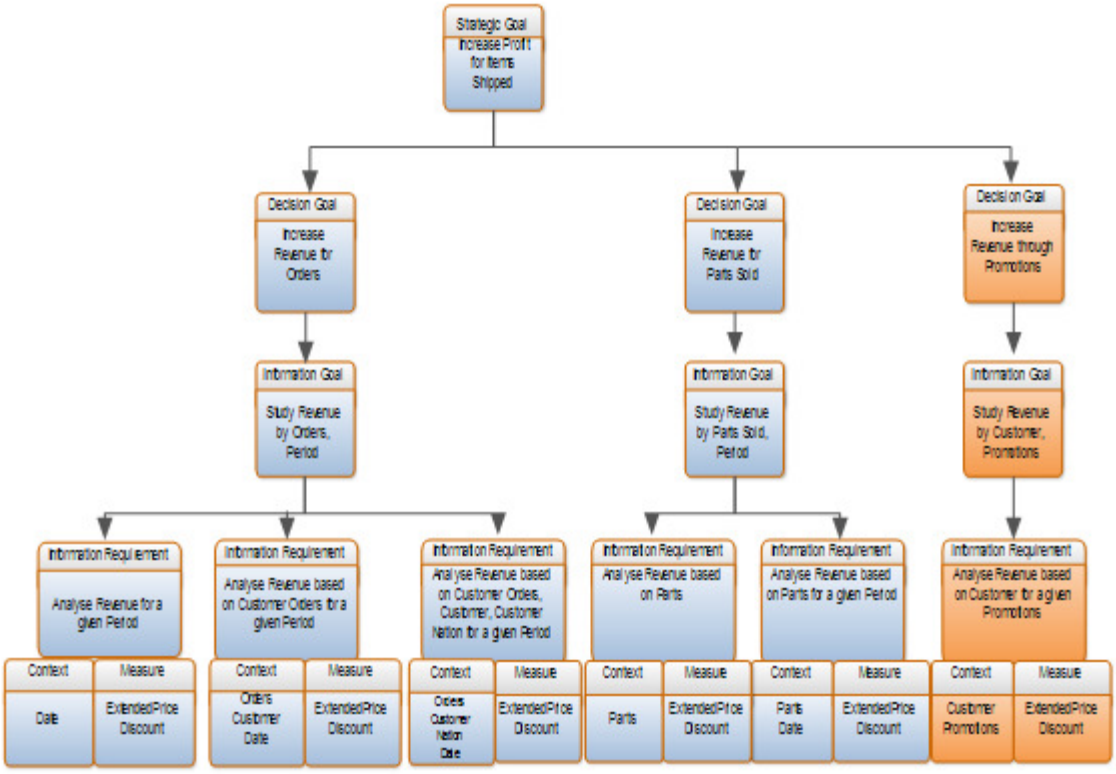

Figure 6. Data Warehouse Requirement Ontology after Updation

Table 2. Change Details.

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. Extract Data Warehouse Schema Change

If change type is addition then it is necessary to find the multidimensional element type of the concept. Figure 7 represents the Algorithm to propagate change. It is used for finding the multidimensional element. The inputs for Algorithm 3 are DSO, change information, multidimensional element list available in DWO.

First the algorithm checks whether the concept added is a class and identifies it as a fact or a dimension or a level. To find whether the class ci is a fact, the range class of ci is obtained. If the range class exists in DSO and it belongs to a dimension in DWO, ci is likely to be a fact. The data properties of ci is derived and checked whether it contains enough numerical properties to qualify as a fact. If ci has n:1 relationship with range class then it is identified as fact (steps 2-16). To find whether ci is a dimension, the domain class of ci is obtained. If the domain class cj belongs to a fact in DWO then ci is likely to be a dimension. If the domain class ci has 1:n relationship with fact then ci is identified as dimension (steps 17-21). To find whether ci is a level, the domain class of ci is obtained. If the domain class cj belongs to a dimension or level in DWO then ci is identified as a level (steps 18-24).

Next the algorithm checks whether the concept added is a data property and identifies it as a fact or a dimension or a level property. The domain d of dpi is obtained. If d is a fact then concept added is identified as fact property. If d is a dimension then concept added is identified as dimension property. If d is a level then concept added is identified as level property.

Finally, the algorithm checks whether the concept added is an object property and identifies it as a fact or a dimension or a level relation. The domain d and range r of opi is obtained. If d is a fact and r is a dimension then concept added is identified as fact-dimension relation. If d is a dimension and r is a fact then concept added is identified as dimension-fact relation. If d is a dimension and r is a level then concept added is identified as dimension-level relation.

In Table 2 the changes related to addition has been listed for the TPC-H domain. By using Algorithm multidimensional type we are able to identify the multidimensional element type for the class “Promotions” and its data properties. Table 3 gives the details of data warehouse multidimensional type of the change that need to be propagated to DWO.

```
Input : Ontology of the form O = {C,DTP,OTP}
Output : MD Concepts: MD = {FL,ML,DL}

FindMDType (Ontology, Concept, Concept _Type)
1       if Concept _Type ==Class then
2               for all ci ∈ C do
3                       if ci == ConceptChanged then
4                               if (Rng(ci.opi)!= null && Rng(ci.opi)∈DimensionList))then
5                                       for all data properties ci .dpi ∈ DP do
6                                               rng := Rng(ci .dpi), rng ∈ DTxml
7                                               if isnumeric(rng) then
8                                                       tn++;
9                                               end if
10                                      end for
11                                      if tn>threshold then
12                                              cj:= Rng(ci.opi), ci ∈ C
13                                              if (ci.opi allValueFrom cj && maxCardinality = = 1) then
14                                                      ci =FactClass
15                                              end if
16                                      endif
17                              else if (Domain(ci.opi)! =null && Domain(ci.opi)∈FactList) then
18                                      cj= Domain(ci.opi)
19                                      if (cj.opi allValueFrom ci && maxCardinality = = 1) then
20                                              ci =DimensionClass
21                                      end if
22                              else if (Domain(ci.opi)! =null && Domain(ci.opi)∈ (DimList || ∈ LevelList)) then
23                                      ci = LevelClass
24                              endif
25                              end if
26                              end if
27                      end if
28              end for
29      else if Concept _Type ==DataProperty then
30              for all dpi ∈ DP do
31                      if dpi == Concept then
32                              d= Domain(dpi)
33                              if d= FactClass then
34                                      dpi=FactProperty
35                              else if d= DimensionClass then
36                                      dpi=DimensionProperty
37                              else
38                                      dpi=LevelProperty
39                              end if
40                              end if
41                              end if
42                      end if
43              end for
44      else if Concept _Type ==ObjectProperty then
45                      for all opi ∈ OP do
46                              if opi == Concept then
47                                      d= Domain(opi)
48                                      r= Rng(opi)
49                                      if (d∈(FactList) && r∈(DimensionList) then
50                                              opi=FactDiemensionRelation
51                                      else if (d∈(DimensionList) && r∈(FactList) then
52                                              opi=DiemensionFactRelation
53                                      else if (d∈(DimensionList) && r∈(LevelList) then
54                                              opi=DiemensionLevelRelation
55                                      end if
56                                      end if
57                                      end if
58                              end if
59                      end for
60      end if
61      end if
62 end if
```

Figure 7. Algorithm Multidimensional Type

Table 3. Multidimensional Type of Addition Change.

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

4.5. Propagate Changes to Data Warehouse

To apply the changes over the DWO, we use different algorithms depending on the type of change. If the type of change is addition the multidimensional element is identified using the previous step. For propagating the addition change we apply Algorithm Apply Change Addition given in Figure 8. If the concept type is a class then we retrieve the list of data properties and object properties for the class to be added. The new class is added to the DWO and for each data property its range and domain is included. Similarly, for each object property its range and domain is included. If the concept type is data property, the new data property is added to the class in DWO and its range and domain are included accordingly. If the concept type is object property, the new object property is added to the class in DWO and its range and domain are included accordingly.

```
ApplyChangeAddition(DWOntology, Concept,Concept _Type)
        if Concept _Type ==Class then
                Add c ∈ C in O
                Get DPList and OPList
                for each dp in DPList
                        Add c .dpi ∈ DP
                        Set Rng(c.dpi)
                        Set Dom(c.dpi)
                end for
                for each op in OPList
                        Add c . opi ∈ OP
                        Set Rng(ci.opi),
                        Set Dom(ci.opi)
                end for
      else if Concept _Type ==DataProperty then
                Get DPList
                for each dp in DPList
                        Add ci .dpi ∈ DP
                        Set Rng(ci . dpi)
                        Set Dom(ci .dpi)
                end for
        else if Concept _Type ==ObjectProperty then
                Get OPList
                for each op in OPList
                        Add c.opi ∈ OP
                                Set Rng(c.opi)
            Set Dom(c.opi)
                end for
        end if
        end if
        end if
```

Figure 8. Algorithm Apply Change Addition

For propagating the deletion change we apply Algorithm Apply Change Deletion given in Figure 9. If the concept type is a class then we delete the class. The corresponding data properties and object properties for the class is also deleted. If the concept type is data property or object property it can be directly deleted from the given class.

```
ApplyChangeDeletion(DWOntology, Concept, Concept _Type)

        if Concept _Type ==Class then
                        Delete c ∈ C in O
                for all data properties c.dpi ∈ DP do
                        Delete  c.dpi
                end for
                for all object properties ci.opi ∈ OP do
                        Delete  c.opi
                                end for
                end if
        if Concept _Type == DataProperty then
                for all data properties c.dpi ∈ DP do
                        Delete  c.dpi
                end for
        end if
        if Concept _Type == ObjectProperty then
                for all object properties ci.opi ∈ OP do
                        Delete  c.opi
                end for
        end if
end if
```

Figure 9. Algorithm Apply Change Deletion

For propagating the rename change we apply Algorithm Apply Change Rename given in Figure 10. If the concept type is a class or data property or object property for the old concept is deleted and the new concept name is included in the ontology.

Table 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 say, “Promotion _p_id”, its corresponding domain and range are included in the DWO. For renaming the Customer dimension data property say, “Customer_c_comment”, its old name is deleted and new name is added as “Customer_c_feedback”. And for deleting the Customer dimension data property say, “Customer_c_mktsegment”, its domain and range are deleted from the DWO. Figure 11 represents the final DWO after the changes are applied.

```
ApplyChangeRename(DWOntology, OldConcept,NewConcept,
Concept _Type)

    if Concept _Type ==Class then
        for all ci ∈ C do
            if ci == OldConcept then
                Add NewConcept
                Delete ci
            end if
        end for
    else if Concept _Type ==DataProperty then
        for all dpi ∈ DP do

            if dpi == OldConcept then
                Add NewConcept
                Delete dpi
            end if
        end for
    else if Concept _Type ==ObjectProperty then
        for all opi ∈ OP do
            if opi == OldConcept then
                Add NewConcept
                Delete opi
            end if
        end for
    end if
```

Figure 10. Algorithm Apply Change Rename

Table 4. Change Propagation

Data Warehouse 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:	Delete Range:
Dimension Data Property	Part_p_container	Delete Domain: Part	Delete Range:

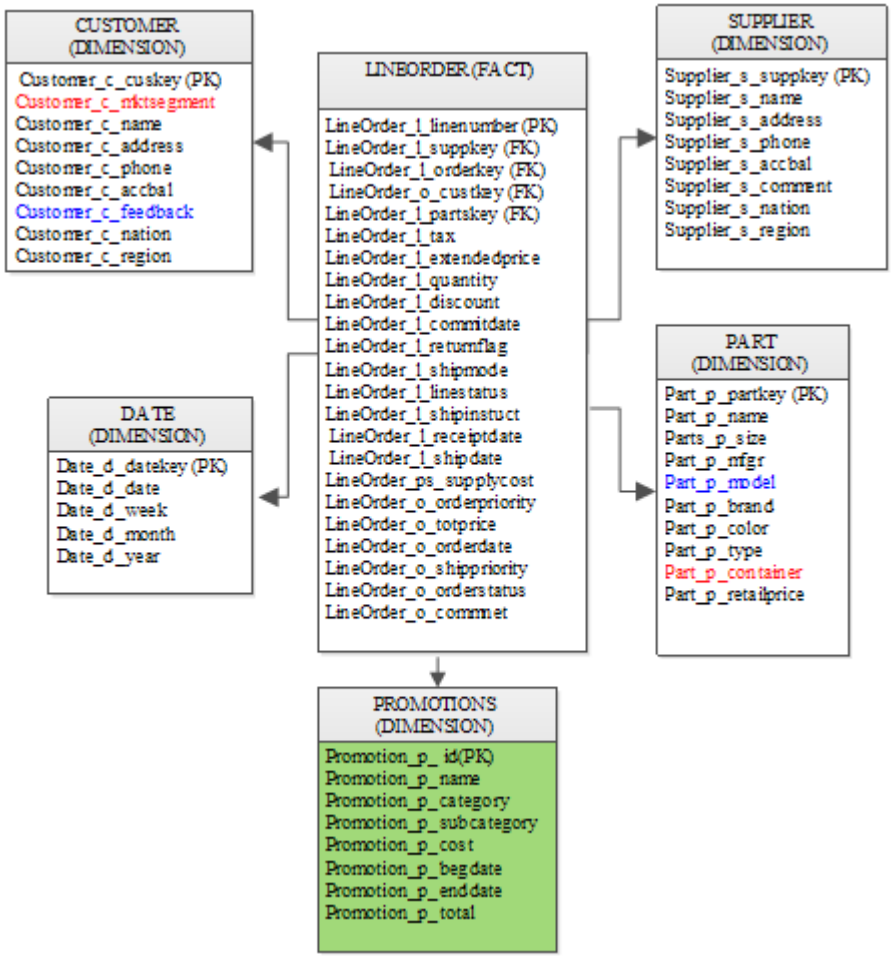

Figure 11. Updated Data Warehouse Ontology

4.6. Check the Consistency

This step is used to check the consistency of DWO after the changes are applied. Ontology reasoner available in protégé [13] is used in order to check the consistency of the given ontology. Following are the steps used check consistency:

- Load the DWO
- Load the reasoner
- Using the loaded reasoner check the consistency of the ontology

Using the DW designer suggestions any inconsistency can be resolved for the DWO. The modified DWO can be kept as a new version of DW schema.

5. EVALUATION OF THE PROPOSED APPROACH

In order to evaluate the efficiency of our approach we examine the cost of manually handling evolution at the physical level with respect to our ontological approach for handling evolution.

The manual effort comprises of detection, inspection and where necessary the rewriting of affected activities by an event.

Human effort for manual handling of schema evolution for a change c, over an event e, is expressed as:

$$MC_c^e = AX_c^e + RX_c^e$$

Where,

AX = no. of changes c, affected by event e, that is manually detected
RX = no. of changes c, which must be manually updated to event e

For a set of evolution operators O, in an activity A, the overall cost of manual adaption to the change c, for an event e is given as:

$$CMA = \sum_{c \in O} \sum_{e \in A} MC_c^e$$

Automatic handling of schema evolution 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 activities AMC that escape the automation A_d, The latter cost AMC is expressed as:

$$AMC = \sum_{c \in O} \sum_{e \in A_d} MC_c^e$$

The overall cost of automated adoption is given by,

$$CAA = CS + AMC$$

The set of evolution operations occurred in the TPC_H source schema included addition of attributes and table renaming of attributes and table, deletion of attributes and table. A total number of 849 evolution operations where encountered and the distribution of occurrence per kind of operation is shown in Figure 12.

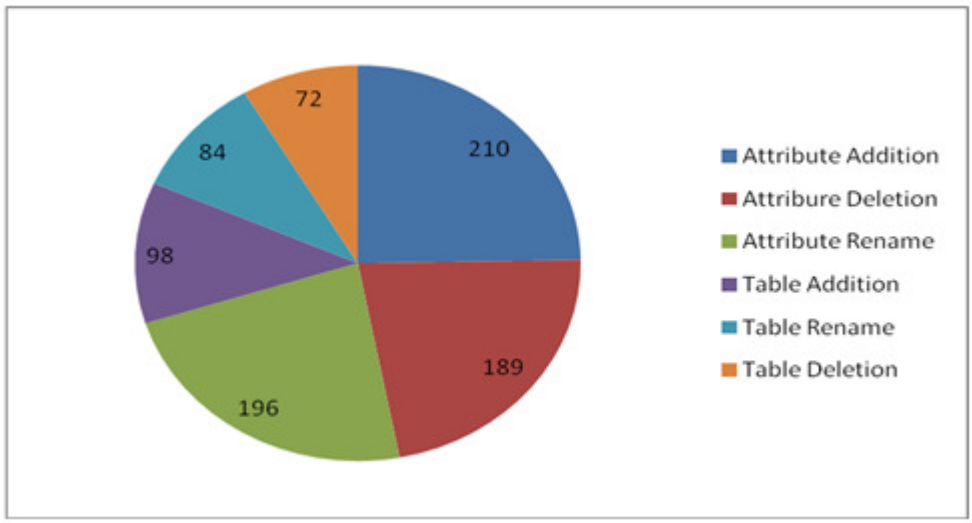

Figure 12. Distribution of occurrence per kind of evolution operations

In Table 4, we summarize our results for different kinds of events. First, we note that most of the activities were affected by attribute additions and renaming, since these kinds of operations were the most common in our scenarios. Most important, we can conclude that our framework can effectively adapt activities to the examined kinds of operations. Figure 13 and Figure 14 shows the comparison of no. of entities affected and that are corrected by using the proposed approach by taking the evolution operators along the x-axis and of no. of entities along the y-axis.

Table 4. Affected and Corrected operations.

Evolution Event Type	Total Affected	Total Corrected
Attribute Addition	210	206
Attribute Deletion	189	189
Attribute Rename	196	194
Table Addition	98	95
Table Deletion	84	83
Table Rename	72	69

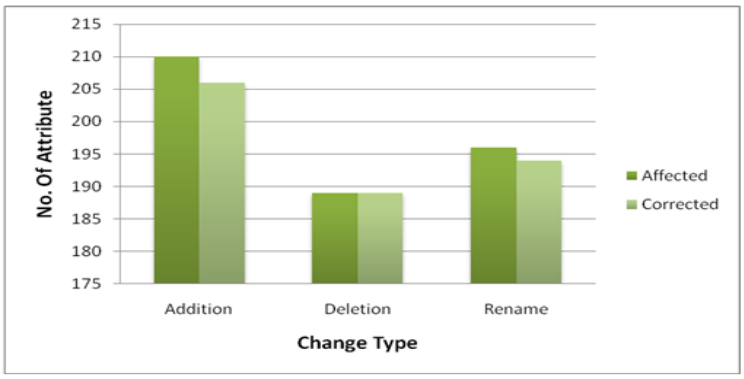

Figure 13. No. of Attributes Affected and Corrected Status

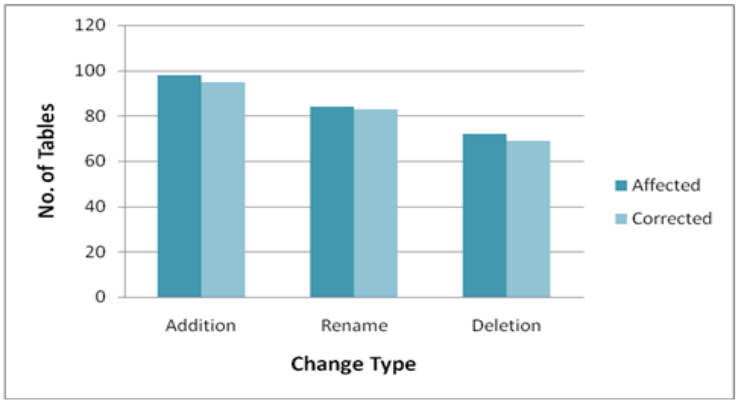

Figure 14. No. of Tables Affected and Corrected Status

From the Figure15 it is found that the cost of automated adaptation (CAA) for the proposed approach is comparatively less than that of manual cost of adoption (CMA) in the existing approach.

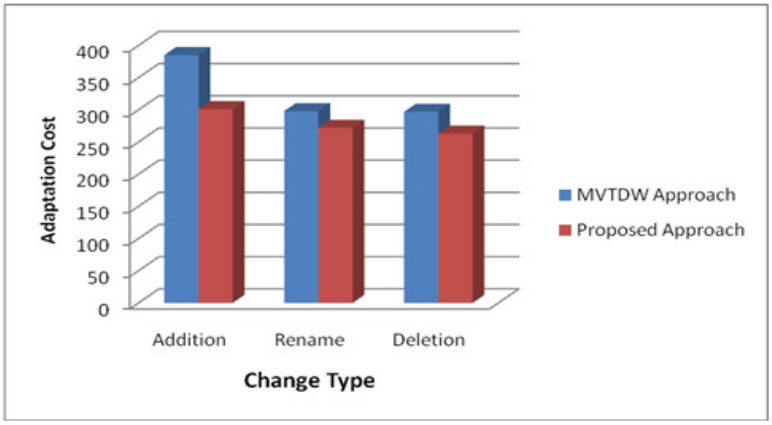

Figure 15. Comparison of Adaptation cost for existing and proposed approach

6. CONCLUSIONS

The data warehouse is considered as the core component of the modern decision support systems. As the information sources and business requirements from which the data warehouse is derived frequently change, it may have its impact on the data warehouse 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 data warehouse schema at the ontological level. The ontological representation of the data source, requirements and data warehouse schema helps us to provide automation (semi-automation) of evolution task. The impact that the evolution has brought over the data warehouse schema are analyzed and the designer is left with the choice of carrying the changes over the existing physical schema of the data warehouse. Compared to existing approaches of manually handling the evolution task the proposed ontological approach provides minimal adaptation cost.

REFERENCES

[1] Banerjee, S., & Davis, K. C. (2009). Modeling data warehouse schema evolution over extended hierarchy semantics. In Journal on Data Semantics XIII (pp. 72-96). Springer Berlin Heidelberg

[2] Bentayeb, F., Favre, C., & Boussaid, O. (2008). A user-driven data warehouse evolution approach for concurrent personalized analysis needs. Integrated Computer-Aided Engineering, 15(1), 21-36.

[3] Chaudhuri, S., & Dayal, U. (1997). An overview of data warehousing and OLAP technology. ACM Sigmod record, 26(1), 65-74.

[4] Chmiel, J., Morzy, T., & Wrembel, R. (2009). Multiversion join index for multiversion data warehouse. Information and Software Technology, 51(1), 98-108.

[5] Council, T. P. P. (2008), TPC-H benchmark specification.[Online] www.tpc.org/tpch/ (Accessed 20 May 2014).

[6] Glorio, O., Pardillo, J., Mazón, J. N., & Trujillo, J. (2008, September). Dawara: An eclipse plugin for using i* on data warehouse requirement analysis.In International Requirements Engineering, 2008. RE'08. 16th IEEE (pp. 317-318). IEEE.

[7] Gruber, T. R. (1993). A translation approach to portable ontology specifications.Knowledge acquisition, Vol. 5 No. 2, pp. 199-220.
[8] Inmon, W. H. (2005). Building the data warehouse. John wiley & so Golfarelli, M., Maio, D., & Rizzi, S. (1998). The dimensional fact model: Aconceptual model for data warehouses. International Journal of CooperativeInformation Systems, 7(02n03), 215-247.
[9] Janet, E., Ramirez, R., Guerrero, E.: A Model and Language for Bi-temporal Schema Versioning in Data Warehouses. (2006). In Proceedings of the 15th International Conference on Computing (CIC '06). IEEE Computer Society, 309-314.
[10] Kimball, Ralph, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, New York, NY: John Wiley and Sons, Inc., 2002. 436pp.
[11] Noy, N. F., & Musen, M. A. (2000, August). Algorithm and tool for automated ontology merging and alignment. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00). Available as SMI technical report SMI-2000-0831.
[12] O'Neil, P., O'Neil, E. J., & Chen, X. (2007). The star schema benchmark [Online] http://www.cs.umb.edu/~poneil/StarSchemaB.PDF (Accessed 20 May 2014).
[13] Ontology, P. (2007). Knowledge Acquisition System. See [Online] http://protege. stanford. edu. (Accessed 12 November 2013).
[14] Oueslati, W., & Akaichi, J. (2010). A survey on Data warehouse evolution. International Journal of Database Management Systems (IJDMS), 2(4), 11-24.
[15] Oueslati, W., & Akaichi, J. (2011). A Multiversion Trajectory Data Warehouse to Handle Structure Changes. International Journal of Database Theory & Application, 4(2).
[16] Smith, M. K., Welty, C., & McGuinness, D. L. (2004). OWL Web OntologyLanguage Guide. W3C. [online] http://www.w3.org/TR/owl-guide/. (Accessed 12 November 2013).
[17] Solodovnikova, D., & Niedrite, L. (2011). Evolution-Oriented User-Centric Data Warehouse. In Information Systems Development (pp. 721-734). Springer New York.
[18] Thakur, G., & Gosain, A. (2011). DWEVOLVE: a requirement based framework for data warehouse evolution. ACM SIGSOFT Software Engineering Notes,36(6), 1-8.
[19] Wrembel, R. (2011). On Handling the Evolution of External Data Sources in a Data Warehouse Architecture.

AUTHORS

M.Thenmozhi received her B.Tech in Computer Science and Engineering from Pondicherry University and M.E in Computer Science and Engineering from Anna University. She is currently pursuing her Ph.D in Computer Science and Engineering, from Pondicherry Engineering College affiliated to Pondicherry University. Presently she is working as Assistant Professor in Department of Computer Science and Engineering, Pondicherry Engineering College. Her research interest includes Data warehousing, Data Modeling, Data mining and Ontology.

Dr.K.Vivekanandan received his B.E from Bharathiyar University, M.Tech from Indian Institute of Technology, Bombay and Ph.D from Pondicherry University. He has been the faculty of Department of Computer Science and Engineering, Pondicherry Engineering College from 1992. Presently he is working as Professor in the Department of Computer Science and Engineering. His research interest includes Software Engineering, Object Oriented Systems, Information Security and Web Services. He has coordinated two AICTE sponsored RPS projects on "Developing Product Line Architecture and Components for e-Governance Applications of Indian Context" and "Development of a framework for designing WDM Optical Network". He has published more than 60 papers in International Conferences and Journals.