ISSN 2229-5518

Implementing Schema Evolution in Data Warehouse through Complex Hierarchy Semantics

Kanika Talwar, Anjana Gosain

Abstract— Data in a data warehouse is collected from several heterogeneous data sources under a unified format, which aims to provide strategic outcomes to the decision makers and facilitate pattern and trend analysis. These data sources are dynamic in nature, due to ongoing transactions in an organization and ever changing requirements. This dynamic nature of the data warehouse has to be dealt with evolution in the data warehouse schema in order to incorporate all the new changes and requirements. In data warehouse systems, the hierarchies play a very important role in processing and monitoring information. So in order to handle complex hierarchies in case of data warehouse evolution, we have proposed evolution operators and certain constraints that need to be fulfilled for ensuring data integrity and schema correctness. This schema correctness in case of evolution is ensured through triggers. In this paper, we have considered a formal metamodel to model the constructs in data warehouse. Also the constraints and operators are defined using the Uni- level Description Language (ULD) and the Multilevel dictionary definition (MDD) approach. The ULD representation exhibits uniform formulation of data, schema and their interrelationships while the MDD structures provide a way for direct implementation in a relational database system.

Index Terms— Data warehouse schema evolution, dimension hierarchies, multi-dimensional schema, evolution operator, constraints, uni- level description, multilevel dictionary definition.

1 INTRODUCTION

A data warehouse is a collection of an enterprise's electronically stored records. It is an integrated, subject-oriented, time-variant and non-volatile compilation of information which supports management's decision making process [1].

Data in the data warehouse is integrated from several autonomous information sources which change or evolve with respect to their data and structure, as a result, data warehouse must also evolve to be preserved in the most up-to-date state. It is however really crucial to consider DW evolution for various reasons.

1. The environment of the data warehouse is dynamic; i.e. ever changing user needs resulting in evolution.
2. Confusing or inadequate requirements during the development phase [2].
3. Periodical revisions done for the removal of bugs & redundancies [3].
4. Change in the information source resulting in new DW design.
5. For the incorporation of new user or requirement in the system or creating new versions [4].

The data warehouse's structure is normally represented with the aid of the multi-dimensional conceptual model. Several examples of which are Star [9], Snowflake [9], Multi-dimensional/ER model [8] and DFM [7]. All these models formulate information as facts and dimensions. A Fact is a numeric value of a normally additive nature which contains measures that are aggregated over dimensions [5]. A dimension comprises of levels and a hierarchy defines a relationship between the levels. This relationship allows data to be viewed at different levels of granularity. Therefore a multidimensional model constitutes facts, dimensions, measures, levels, and hierarchies which exhibit subsequent features:

1. Facts and dimensions having a many-to-one relationship i.e. a dimension instance corresponds to many fact instances.
2. Different levels of a dimension having a many-to-one roll-up relationship.
3. Hierarchies in a dimension having one-to-many drill-down relationship i.e. one parent and many child.

Banerjee [6] referred these constraints as the core features of DW conceptual model.

Several authors such as Tsois [10], Pedersen [11] and Hummer [12] have discussed about the different problems that arise in real world business application domains and the inadequacy of different traditional conceptual models to handle them. Banerjee and Davis [6] handled some of those problems like multiple hierarchies, Non-onto (missing data), Non-covering hierarchies and Non-strict hierarchy (many-to-many relation between parent and child level) with the help of a new formal metamodel accompanying ULD [14, 15,16] and MDD approach [13]. But some important extended hierarchies are not covered by [6]. In our work we have considered the following complex extended hierarchies [17] (given in table 1):

- *Kanika Talwar is currently pursuing M.tech. in Computer science & engineering in USICT, Guru Gobind Singh Inderprastha University, Delhi, India, E-mail: kanika.ncce@gmail.com*
- *Dr. Anjana Gosain is currently working as reader in CSE/IT department in USICT, Guru Gobind Singh Inderprastha University, Delhi, India. E-mail: anjana_gosain@hotmail.com.*

TABLE 1
EXTENDED HIERARCHIES

S.No	Hierarchies	Description
1.	Multiple Alternative hierarchy	A dimension contains several non-exclusive paths sharing some levels and accounting for the same analysis criterion.
2.	Parallel Dependent hierarchy	A dimension has different simple hierarchies sharing some levels and accounting for different analysis criterion.
3.	Parallel Independent hierarchy	A dimension contains different simple hierarchies which do not share any level and account for different analysis criterion.

To model the above extended hierarchies we have merged some examples from the literature and formed a schema shown in Figure 1.We have used modified DFM [7] notation; where the first level in each dimension is same as the dimension name. Also the finer-granule level like Brand rolls up to the coarser-granule level, i.e. Corporation in the Product dimension. In the example schema (Fig 1), the Sales fact table is taken from Golfarelli [18] paper having measures Quantity Sold, Revenue, and No. of Customers. The second example is presented by Hurtado [19] i.e. Product dimension which exhibits multiple alternative hierarchies as the level Corporation drills-down to Company or Category. The two paths of the dimension that portrays the roll-up from the lowest to the highest hierarchy level are Category, Corporation and Brand, Company, Corporation. Both the paths converge at same level. The next example i.e. Store dimension is presented by Malinowski [17] which exhibits Parallel Dependent hierarchy.

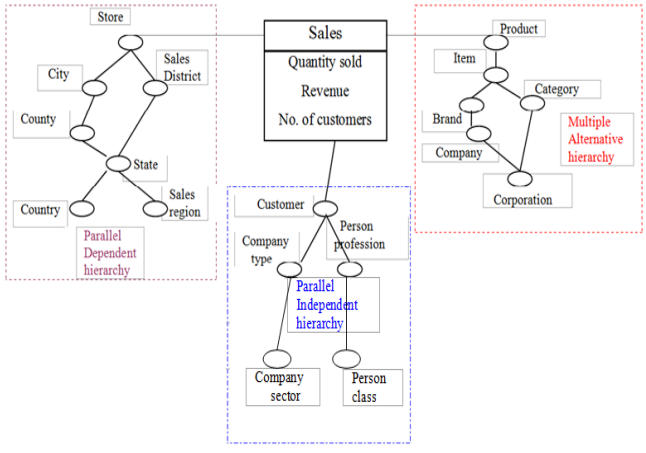

Fig. 1. Sales Schema

There are two different analysis criterions; 1) city, county, state and country and 2) sales district, state and sales region in Store dimension. Both the paths share a common level i.e. State. The Customer dimension exhibits parallel independent hierarchy where it can be analysed by two different analysis criterion; Company (type and sector) and Person (profession and class).

In the literature, several authors have proposed operators to handle schema evolution at different levels focusing on core as well as extended data warehouse features. Bouzeghoub [21] and Quix [22] focus on addition/deletion of views. Blaschka [20] presents a formal model conceptually similar to the metamodel [6] used by us and use algebra to define fact and dimension evolution. Hurtado [19] focus on changes to dimensions only. Golfarelli [23] focuses on evolution of hierarchies within a dimension. Chen [24] has operators for adding, deleting, and renaming tables and attributes. None of the above DW schema evolution papers propose a tool for implementing changes as done by Banerjee [6]. The schema evolution operators defined by [6], focuses on schema correctness. The tool creates the data warehouse schema rapidly and checks the validity of schema evolution operations thereby guarantying schema correctness. It has also considered some additional features like multiple, non-strict hierarchies, non-covering hierarchies etc. But it has not distinguished between various variations of multiple hierarchies like multiple alternative, parallel dependent hierarchies etc. In this paper, in addition to basic structural constructs of data warehouse model, we are also going to define constraints separately for above hierarchies and propose evolution operator on them. The constraints must be satisfied for schema correctness, which is enforced by triggers.

Here we are using ULD definition along with multilevel dictionary definition approach [13] as a tool to express our model and evolution operators for the schema. ULD definition exhibits formal semantics and uniform representation of schema data, metamodel layers and their inter-dependencies, thereby making modeling easier. While the MDD structures allow direct implementation in a relational database system, thereby giving a basis for ensuring schema correctness.
In Section 2, we discuss about the formal metamodel constructs and constraints for extended hierarchy semantics. In Section 3, we demonstrate the multilevel dictionary implementation of the model and use the Sales schema of Figure 1 to exhibit schema evolution operations. Finally, Section 4 gives conclusion and future perspectives.

2 BACKGROUND, CONTRIBUTION AND RELATED NEW WORK

Bowers and Delcambre presented the Uni-Level Description language [14,15,16] which provides a formal framework for representing a wide range of data models such as ER, XML and RDF. It is considered as a generic representing language capable of enabling powerful transformation rules as well as simple generic browsing capability over the information that is originated within various data models and representation

schemes. The main advantage of using ULD in describing formal metamodel (set of constructs) is that, it supports data model interoperability which facilitates easy and universal information access in case of multiple heterogeneous data sources. It is termed as the uni-level representation language because it completely describes the data model, its sources, the schemas present and all the instance data in a single flat representation [16].

In this paper, we also bring into use Multilevel Dictionary Definition (MDD) approach [13] along with ULD to represent the model [6] into schema constructs and define schema operators for data warehouse evolution. MDD allows direct implementation of data model constructs in the form of tables which makes it easy to understand. The constructs are implemented in a relational database system. We use triggers for the implementation of constraints and written an algorithm for evolution operator which thereby enforce semantics. To make changes in any of the schema construct, we require constraints to ensure correctness of the model. This change (add/delete) is done with the help of evolution operators. The very first step is to define the schema constructs along with semantics using ULD, followed by representation of constructs in MDD and at last, implementing the constructs, constraints and operators in relational databases by means of triggers.

Here, we are using two of the ULD construct types i.e. setct and structct [16], where setct corresponds to a set of objects and structct stands for a structured object with subcomponents. The construct type structct is denoted as "construct c := [a => b]", where "a => b" denotes a component of the construct c, a is termed as the component selector and b is the type of the component. The setct construct type is expressed as set-of. For instance, a DimensionSet is represented as a set-of Dimension, where a Dimension is a structct construct having subcomponents as Dname, Dlevel, DPkey, and Dhierarchy and Dpath fields [6]. A Dpath construct is of the type PathSet, which defines a set of paths in a dimension.

So here now, we define a new schema construct called the '**Path'** construct which is a set-of **PathLevels** in a dimension (given in fig. 2), instead of PathHierarchies as defined by Banerjee [6]. The rest of the schema constructs are same as stated by [6].

- construct Dimension := [Dname => uld-string, Dlevel => LevelSet, DPkey => uld-string, Dhierarchy => HierarchySet, Dpath => PathSet]:: Fact cardinality [1:n]
- ***construct PathSet := set-of Path***
- ***construct Path := set-of PathLevel***

Fig. 2. Path Construct in Uni-Level Description Language

The correctness and consistency of the schema constructs in the model are sustained by inserting certain addition/deletion constraints. So here now we introduce addition constraints for the path construct considering various hierarchies discussed in table 1. Constraints C.1 C.2 and C.3 (given below) enforce the semantics for new paths to be added in the path construct of respective hierarchies. For instance, constraint C.1 can be explained in other words as: Multiple alternative hierarchies consist of several non-exclusive paths sharing some levels. It is required that the two paths converge at same level i.e. they should share a level which is the most coarser-grained level (having no parent) E.g. Corporation in figure 1.

1. Multiple Alternative hierarchy

C.1 A new path added to the dimension construct should converge at the same level as of the previous path in the same dimension.

Addition Constraint (C.1)
∀ (a,b,c,d) ∃ (e) c-inst(a, Dimension) ∧ member-of(Dlevel, a) ∧ c-inst(b, Dlevel) ∧ IsHierarchy(c, a) ∧ member-of(ChildLevel, c) ∧ c-inst(d, Dpath) ∧ member-of(Plevel, d) ∧ c-inst(e, Plevel) ∧ addDpath(d, a) → ¬ c-inst(b, ChildLevel) ∧ equivalent(e, b)
Explanation
For all a,b,c and d there exists some e, where a is an instance of Dimension construct and Dlevel is the set of levels in a, and b is an instance of Dlevel in Dimension a, and c is a hierarchy in Dimension a, and d is a new path added to Dimension a, and e is an instance of Pathlevel in path d, then b is not a child level in the hierarchy c and e is equivalent to b.

2. Parellel Dependent hierarchy

C.2 A new path added to the dimension construct should have at least one level in common to previous path in the same dimension.

Addition Constraint (C.2)
∀ (a,b) ∃ (c) c-inst(a, Dimension) ∧ member-of(Dlevel, a) ∧ c-inst(b, Dpath) ∧ member-of(Plevel, b) ∧ c-inst(c, Plevel) ∧ addDpath(b, a) → c-inst(c, DLevel)
Explanation
For all a,b there exists some c, where a is an instance of Dimension construct and Dlevel is the set of levels in a, and b is a new path added to Dimension a, and c is an instance of Pathlevel in path b, then there exists atleast one level in the new path equivalent to Dimension level.

3. Parallel Independent hierarchy

C.3 A new path added to the dimension construct should not have any common level to previous path in the same dimension.

Addition Constraint (C.3)
∀ (x,y) ∃ (z) c-inst(x, Dimension) ∧ member-of(Dlevel, x) ∧ c-inst(y, Dpath) ∧ member-of(Plevel, y) ∧ c-inst(z, Plevel) ∧ addDpath(y, x) → ¬ c-inst(z, DLevel)
Explanation
For all x,y there exists some z, where x is an instance of Dimension construct and Dlevel is the set of levels in x, and y is a new path added to Dimension x, and z is an instance of Pathlevel in path y, then there exists no level in the new path equivalent to Dimension level.

The following section demonstrates the implementation of the schema evolution using multilevel dictionary definition approach along with an example.

3 Implementation of Schema Evolution using MDD

The Multilevel Definition dictionary approach proposed by Atzeni et. al. [13] is referenced here to represent the model constructs in the form of tables. An example to demonstrate schema evolution with MDD approach is explained below considering the three dimensions i.e. product, location and customer of figure 1. The tabular format of constructs Dimension, Level, Hierarchy and Path are given in table 2, 3, 4 and 5 respectively.

TABLE 2
DIMENSION CONSTRUCT

S. No	DName	DLevel	DPkey	DHierarchy	Dpath
D1	Product	L1,L2,L3,L4,L5	Product#	h1,h2,h3,h4,h5	p1,p2
D2	Customer	L6,L7,L8,L9	Customer#	h6,h7	p3,p4
D3	Store	L10,L11,L12,L13,L14, L15	Store#	h8, h9,h10,h11,h12	p5,p6

TABLE 3
PATH CONSTRUCT

S.No.	PLevel
p1	L1,L2,L3,L5
p2	L1,L4,L5
p3	L6,L7
p4	L8,L9
p5	L10,L11,L13,L14
p6	L12,L13,L15

TABLE 4
LEVEL CONSTRUCT

S.No	Level Name
L1	Item
L2	Brand
L3	Company
L4	Category
L5	Corporation
L6	Company type
L7	Company sector
L8	Person profession
L9	Person class
L10	City
L11	County
L12	Sales district
L13	State
L14	Country
L15	Sales region

TABLE 5
HIERARCHY CONSTRUCT

S.No.	Parent Level	Child Level
h1	L2	L1
h2	L3	L2
h3	L5	L3
h4	L4	L1
h5	L5	L4
h6	L7	L6
h7	L9	L8
h8	L11	L10
h9	L13	L11
h10	L14	L13
h11	L13	L12
h12	L15	L13

Evolving the schema means making changes in the table records. These changes are done with the help of operators. An example schema evolution operator is represented by an algorithm given in figure 3. These updations of records thereby invoke the triggers to be applied on the tables which in case enforce semantics. A trigger on a database can be defined as a procedural code which is invoked on its own, in case of any modification in a table of the database. For instance, if a new path is added to the dimension construct of the multiple alternative hierarchy, then the coarser granular level should be the shared level between the new and the old path of the same dimension. A sample trigger based on the addition constraint C.2 (defined in section 2) is given in figure 4.This trigger is created over the path construct. If a new path is added to the

dimension using AddPath_PD operator, then if there is no common level between the new path and the existing paths; the transaction is rolled back.

Algorithm AddPath_PD

Input: Schema Name, Dimension Name, PLevel
Output: A new path is added to the existing dimension with one shared level.

Step 1: Check if Schema Name is correct.
Step 2: Check if Dimension Name is a valid dimension in the schema.
Step 3: Check if at least one level in new path is an existing level of the dimension.
Step 4: If above 3 steps are correct, then add new path to the DPathSet Construct.

Fig. 3. Algorithm for AddPath Evolution Operator

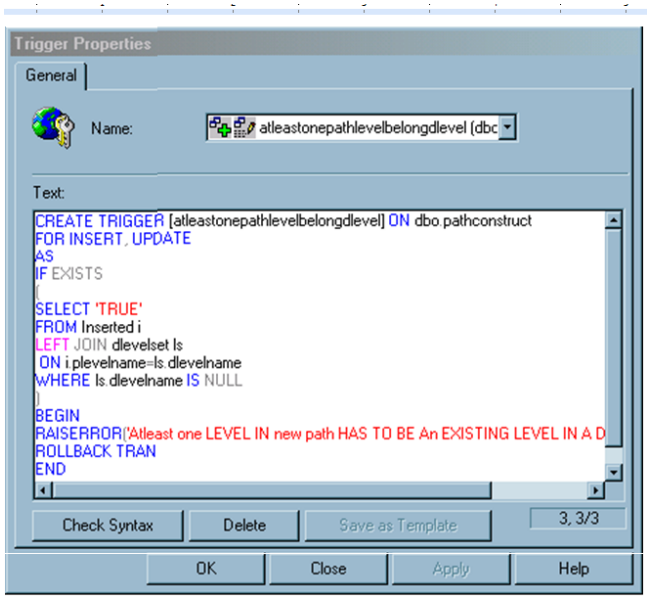

Fig. 4. A Sample trigger on path construct

Here we defined a data warehouse schema and applied constraints on it for correct evolution. We used the SQL Server environment to implement the constraints. The evolution operators are applied to evolve a schema by altering its tables and triggers help in keeping the consistency intact. Moreover the use of ULD and MDD approach together made it possible to model the schema evolution using extended hierarchies.

4 Conclusions and Future Work

The main focus of the paper is to handle the schema evolution in the data warehouse caused due to ever changing requirements and therby improving the analysis process in the data warehouse. In order to handle schema evolution of certain extended hierarchies prevailing in the data warehouse, we take into account three hierarchies namely Multiple alternative, Parallel dependent and Parellel independent hierarchies and defined constraints (C.1, C.2, C.3) for it that need to be satisfied for enforcing semantics and schema correctness. We also proposed an algorithm for the evolution operator for Parellel dependent hierarchy. Our future work includes handling of generalized hierarchies in schema evolution. Also we aim to support cross- version quering in the used model.

References

[1] Inmon, W.: Building the Data Warehouse, pp 23 (1991).
[2] Body, M., Miquel, M., Bedard, Y., Tchounikine, A.: Multidimensional and Multiversion Structure for OLAP applications: In Proc. of the 5th ACM Intl. Workshop DOLAP, 1-6 (2002).
[3] Bebel, B., Krolikowski, Z., Wrembel, R.: Formal Approach to Modeling Data Warehouse. In bulletin of the Polish Academy of Sciences, 54, 1 (2006).
[4] Janet, E., Ramirez, R., Guerrero, E.: A Model and Language for Bitemporal Schema Versioning in Data Warehouses: In Proc. of 15th International Conference on Computing (2006).
[5] Multi-Dimensional Modeling with BI, Version 1.0, May 16, 2006: ©2000 SAP AG and SAP America, Inc.
[6] S.Banerjee, K.C.Davis, Modeling Data Warehouse Schema Evolution over Extended Hierarchy Semantics, S.Spaccapietra et.al (EDs): Journal on Data Semantics XIII, LNCS 5530, pp.72-96,2009. @Springer-Verlag Berlin Heidelberg 2009.
[7] Golfarelli, M., Maio, D., Rizzi, S.: The Dimensional Fact Model: A Conceptual Model for Data Warehouse: International Journal of Cooperative Information Systems (IJCIS) 7(2-3), 215–247 (1998).
[8] Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Kambayashi, Y., Lee, D.-L., Lim, E.-p., Mohania, M., Masunaga, Y. (eds.) ER Workshops 1998. LNCS, vol. 1552, pp. 105–116, Springer, Heidelberg (1999).
[9] Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology, SIGMOD Record 26(1), 65–74 (1997)
[10] Tsois, A., Karayannidis, N., Sellis, T.: MAC: Conceptual data modeling for OLAP. In: Proceedings of the 3rd International Workshop on Design and Management of Data Warehouses (DMDW), Interlaken, Switzerland, June 4, p. 5 (2001).
[11] Pedersen, T., Jensen, C.: Multidimensional Data Modeling for Complex Data. In: Proceedings of 15th International Conference on Data Engineering (ICDE), Sydney, Australia, March 23-26, pp. 336–345 (1999)
[12] Hümmer, W., Lehner, W., Bauer, A., Schlesinger, L.: A decathlon in multidimensional modeling: Open issues and some solutions. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 275–285. Springer, Heidelberg (2002)
[13] Atzeni, P., Cappellari, P., Bernstein, P.: A multilevel dictionary for model management. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 160–175. Springer, Heidelberg (2005)
[14] Bowers, S., Delcambre, L.: On Modeling Conformance for Flexible Transformation over Data Models, Knowledge Transformation for the Semantic Web 95, 34–48 (2003).
[15] Bowers, S., Delcambre, L.: The uni-level description: A uniform framework for representing information in multiple data models. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003.

LNCS, vol. 2813, pp. 45–58. Springer, Heidelberg (2003)

[16] Bowers, S., Delcambre, L.: Using the Uni-Level Description (ULD) to Support Data-Model Interoperability. Data and Knowledge Engineering 59(3), 511–533 (2006)

[17] E. Malinowski, E. Zimanyi, "Hierarchies in a multidimensional model: From conceptual modeling to logical representation," Data & Knowledge Engineering 59 (2006) 348–377.

[18] Golfarelli, M., Rizzi, S.: A Methodological Framework for Data Warehousing Design. In: Proceedings of the 1st International Workshop on Data Warehousing and OLAP (DOLAP), Washington, DC, USA, November 2-7, pp. 3–9 (1998)

[19] Hurtado, C., Mendelzon, A., Vaisman, A.: Maintaining Data Cubes under Dimension Updates. In: Proceedings of 15th International Conference of Data Engineering (ICDE), Sydney, Australia, March 23-26, pp. 346–355 (1999)

[20] Blaschka, M., Sapia, C., Höfling, G.: On schema evolution in multidimensional databases. In: Mohania, M., Tjoa, A.M. (eds.) DaWaK 1999. LNCS, vol. 1676, pp. 153–164. Springer, Heidelberg (1999)

[21] C. Quix. Repository Support for Data Warehouse Evolution,In Proc. of the Intl workshop DMDW, Heidelberg, Germany 1999.

[22] Bouzeghoub, M., and Z. Kedad, "A Logical Model for Data Warehouse Design and Evolution," Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK), London, UK, September 4-6, 2000, pp. 178-188.

[23] Golfarelli, M., Lechtenbörger, J., Rizzi, S., Vossen,G.: Schema Versioning in Data Warehouses. In: Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D.Grandi, F., Mangina, E.E., Song, I.-Y., Mayr,H.C. (eds.) ER Workshops 2004. LNCS, vol. 3289, pp. 415–428. Springer, Heidelberg (2004)

[24] Chen, J., Chen, S., Rundensteiner, E.: A transactional model for data warehouse maintenance. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 247–262. Springer, Heidelberg (2002)