Web-Based Implementation of Data Warehouse Schema Evolution

Jai W. Kang, Fnu Basrizal, Qi Yu, and Edward P. Holden

Rochester Institute of Technology
152 Lomb Memorial Drive, Rochester, NY, USA
{jai.kang,fxb3717,qi.yu,edward.holden}@rit.edu

Abstract. An organization collects current and historical data for a data warehouse from disparate sources across the organization to support management for making decisions. The data sources change their contents and structure dynamically to reflect business changes or organization requirements, which causes data warehouse evolution in order to provide consistent analytical results. This evolution may cause changes in contents or a schema of a data warehouse. This paper adapts a schema evolution method to address the data warehouse evolution given a data warehouse is built as a multidimensional schema. While existing works have identified and developed schema evolution operations based on conceptual models to enforce schema correctness, only a few have developed software tools to enforce schema correctness of those operations. They are also coupled with specific DBMSs (e.g., SQL Server) and provide limited GUI capability. This paper aims to develop a web-based implementation to support data warehouse schema evolution. It offers all the benefits that Internet browser-based applications provide, by allowing users to design, view, and modify data warehouse schema graphically. This work focuses on evolution operations in dimensional tables, which includes changes in levels, hierarchies, and paths. Schema correctness for each schema evolution operation is ensured by procedural codes implemented in PHP.

Keywords: data warehouse, schema evolution, web-based application, dimensional modelling, data mart, star schema.

1 Introduction

An organization develops and maintains a data warehouse to support management for making decisions. The data warehouse houses both current and historical transactional data from various data sources across the organization. A data warehouse can be a large normalized enterprise-wide relational database from which de-normalized multidimensional databases can be generated for end users to understand and write queries easily against.

While data marts can be generated from the enterprise data warehouse, modeling as a star schema for each business process can develop a data mart independently. The data mart as a dimensional model classifies data as facts and dimensions. Facts contain numeric data as a result of a business, called a measure. Dimensions contain descriptive information to filter and aggregate the facts. Kimball & Ross [4] advocate that an enterprise data warehouse is a union of data marts as long as all data marts are conformed to following the same convention and standards, which is called a bus architecture.

A dimension consists of mostly textual attributes that can be characterized as levels and hierarchies. Malinowski & Zimányi [5] state that the level allows data warehouse users to explore the measures from a different perspective of analysis. The levels may create a hierarchy so that users can access the measures in detailed or generalized information. The hierarchy is defined as a relationship between two levels. A path is defined as a set of hierarchies in a dimension table. A dimension can have multiple paths for alternative or different analysis criteria. A path represents a specific analysis criterion.

Data of a data warehouse come from many different data sources. These data sources tend to change their contents and structure dynamically to reflect business changes or organization requirements. Changes in contents and structures of data sources may cause schema changes in a data warehouse [3]. Therefore, data warehouse evolution can be caused by the dynamic nature of data sources. The evolution may cause changes in content or a schema of a data warehouse. Slowly changing dimension (SCD) as proposed by Kimball & Ross [4] handles changes that occur for the content of data warehouse. For schema changes, there are three different methods to address a data warehouse evolution: (1) schema evolution, (2) schema versioning when a data warehouse is built as a multidimensional schema and (3) view maintenance when a data warehouse is built as a collection of materialized views [6].

Schema change operations require taking into account hierarchies in a data warehouse in order to ensure that the schema changes do not violate existing hierarchies. Hierarchy has an important function to process information that allows users to view and explore data in a data warehouse at various levels of granularity. Specifically, using hierarchy, a user can view data from a general view to a detailed view through roll-up, drill-down, slicing and dicing operations.

This paper aims to handle schema evolutions that occur in dimension tables of a data warehouse. Schema evolution can be structure changes in levels, hierarchies, and paths of dimension tables. The proposed approach implements a web-based schema evolution that performs and validates schema evolution operations to achieve schema correctness in a data warehouse. A web-based application offers key benefits over platform specific ones for obvious reasons, as users of a web-based system do not require having their computers equipped with particular software and hardware, but an Internet browser.

The rest of the paper is organized as follows. Section 2 presents dimension hierarchical categorization and classification, and existing works on the implementation of data warehouse schema evolution. Web-based implementation is given by adopting a Multi-level Dictionary Definition (MDD) approach in Section 3. Finally, Section 4 concludes the paper and discusses future work.

2 Background, Related Works and Motivation

2.1 Dimensional Hierarchies

A data warehouse offers multidimensional models for users to analyze a large volume of data. A dimensional model classifies data as a fact and dimensions, which are modeled as a star schema. Users can then explore the facts in various perspectives using the hierarchical nature of dimension attributes. These hierarchies allow users to view and explore data at various levels of granularity. Using hierarchy, a user can view data from a general view to a detailed view through operations like roll-up and drill-down.

Talwar & Gosain [7] recognize the necessity of properly categorizing dimension hierarchies so as to properly model them during evolution. They define a hierarchy as a set of binary relationships existing between dimension levels, where a dimension level participating in a hierarchy is called a hierarchical level. Based on two consecutive levels of a hierarchy, the higher level is called *parent* and the lower level is called *child*. The *leaf* level is the finest level in a hierarchical path, which has no child; the last level, i.e., the one that has no parent level called *root* level. A hierarchy represents some organizational, geographical, or other type of structure that plays a key role for analysis [5].

Talwar & Gosain [7] performs a literature survey on hierarchical classification of data warehouse proposed by various researchers based on certain parameters from a wide range of business scenarios. Their classifications include ones by Malinowski & Zimányi [5] and Banerjee et al. [2] such as simple, multiple, parallel dependent and independent hierarchies. These hierarchies are described in Table 1, which also includes non-strict, non-onto and non-covering hierarchies defined by Talwar & Gosain [8].

2.2 Related Works

Existing works on the implementation of data warehousing schema evolution include Banerjee and Davis [2] and Talwar and Gosain [7]. Both works define a formal meta-model for data warehouse core features using Uni-Level Description (ULD) language and a Multi-level Dictionary Definition (MDD). The core features of a data warehouse conceptual model refers to a multidimensional model consisting of facts, dimensions, measures, levels and hierarchies that conform to: 1) a many-to-one relationship between a fact and a dimension, 2) a one-to-many relationship between two levels in a dimension, and 3) hierarchies in a dimension have a single path to roll-up or drill-down operations. While the ULD definition provides formal semantics and uniform representation of schema data, meta-model layers and their inter-dependencies, the MDD allows direct implementation in a relational database system. Banerjee and Davis [2] implemented such advanced features as multiple hierarchies, non-covering, non-onto, and non-strict hierarchies. Talwar and Gosain [7] implemented such extended hierarchies as multiple alternative, parallel dependent, and parallel independent hierarchies. Both works use stored procedures and triggers to enforce schema correctness in Microsoft SQL Server.

Table 1. Dimension hierarchies [7]

Hierarchy	Description
Simple	This hierarchy can be represented as tree. They use only one criterion for analysis
Multiple	This hierarchy contains several non-exclusive simple hierarchies and share some levels, but all of these hierarchies have the same analysis criterion
Parallel	Parallel hierarchies arise when there are multiple hierarchies, accounting for different analysis criteria. Two types: Parallel independent/dependent hierarchy depending sharing any level(s) or not
Non-strict	This exists when a dimension can have many-to-many relationships
Non-onto	This hierarchy exists when lower level can exist without a corresponding data in the higher level to roll-up to.
Non-covering	This exists when at least one member whose logical parent is not in the level immediately above the member

2.3 Motivation

Users of these existing applications perform data warehouse schema evolution on a specific DBMS, such as Microsoft SQL Server. The system offers a simple GUI that prompts users to supply arguments or using the SQL command line in order to execute stored procedures.

The paper aims to make a key extension of existing approaches by moving from platform specific applications to a web-based system. The benefits of a web-based system can be related to minimizing software installation, update and training among others. Google search returns over 140 million hits for the query "Benefits of web-based application" as of June 2014. Furthermore, [9] includes "Enterprise applications migrating to browsers" in Gartner's 2014 top ten technology trends. In fact, such a trend has already been observed during the past decades in the application development industry.

3 Web-Based Implementation of Schema Evolution Using MDD

3.1 Implementation Example

This project adopts the MDD approach of schema evolution developed by Atzeni et al. [1], which manages schema and describes its components. It follows the same approach for designing the data model constructs as [8] and [2]. It uses MDD to represent core features of a data warehouse and meta-constructs of ULD instead of the supermodel proposed by Atzeni et al. [1]. The constructs are implemented in a relational database system.

Fig. 1 shows an example of data warehouse schema to demonstrate schema evolution using the MDD approach. The example is taken from Banerjee & Davis [2] and Talwar & Gosain [7]. The schema shows four dimensions: product, customer, store and location. Location dimension here is used specifically to show an implementation of non-covering, non-strict and non-onto hierarchies. The tabular format of constructs includes Schema, Fact, Primary Key and Measure, but they are not shown here due to space restriction. Tabular constructs of Dimension, Level, Hierarchy, Path, Non-Covering, Non-Onto and Non-Strict are given in Tables 2 through 8 respectively.

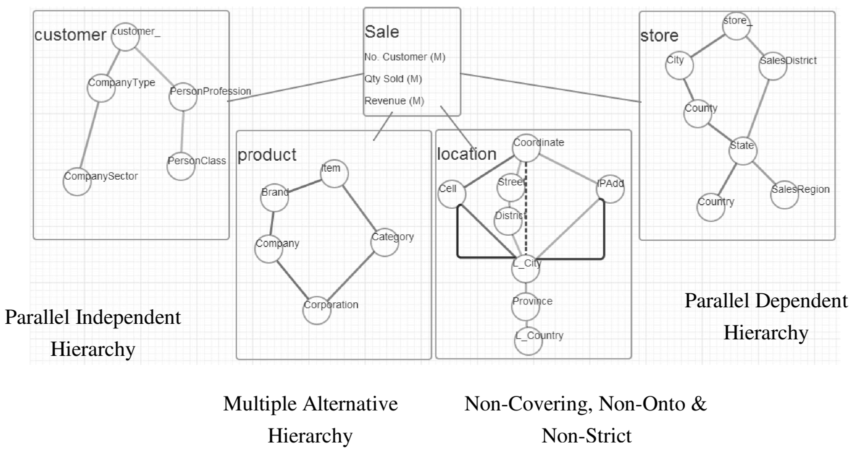

Fig. 1. Sales schema

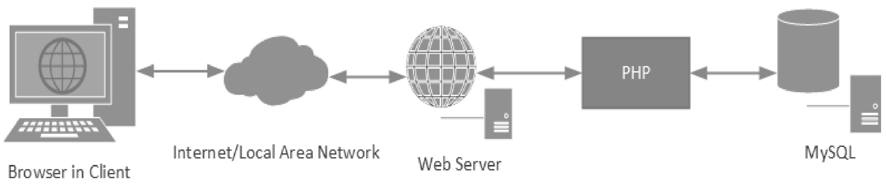

Fig. 2. Application architecture

Table 2. Dimension construct

Id	Name	DLevel	DPKey	DHierarchy	Dpath
d1	Product	L1,L2,L3,L4,L5	pk1	h1,h2,h3,h4,h5	p1,p2
d2	Customer	L6,L7,L8,L9,L10	pk2	h6,h7,h8,h9	p3,p4
d3	Store	L11,L12,L13, L14,L15,L16,L17	pk3	h10,h11,h12,h13, h14,h15,h16	p5,p6
d4	Location	L18,L19,L20,L21, L22,L23,L24,L25	pk4	h17,h18,h19,h20, h21,h22,h23,h24h25	p7,p8,p9

Table 3. Level Construct

Id	Name
L1	Item
L2	Brand
L3	Company
L4	Category
L5	Corporation
L6	Customer
L7	Company Type
L8	Company Sector
L9	Person Profession
L10	Person Class
L11	Store
L12	City
L13	County
L14	State
L15	Country
L16	Sales District
L17	Sales Region
L18	Coordinate
L19	Street
L20	District
L21	L_City
L22	Province
L23	L_Country
L24	IPAdd
L25	Cell

Table 4. Hierarchy Construct

Id	Parent Level	Child Level
h1	L2	L1
h2	L3	L2
h3	L5	L3
h4	L4	L1
h5	L5	L4
h6	L7	L6
h7	L8	L7
h8	L9	L6
h9	L10	L9
h10	L12	L11
h11	L13	L12
h12	L14	L13
h13	L15	L14
h14	L16	L11
h15	L14	L16
h16	L17	L14
h17	L19	L18
h18	L20	L19
h19	L21	L20
h20	L22	L21
h21	L23	L22
h22	L24	L18
h23	L21	L24
h24	L25	L18
h25	L21	L25

Table 5. Path Construct

Id	Phierarchy
p1	h1,h2,h3
p2	h4,h5
p3	h6,h7
p4	h8,h9
p5	h10,h11,h12,h13
p6	h14,h15,h16
p7	h17,h18,h19,h20,h21
p8	h22,h23,h20,h21
p9	h24,h25,h20,h21

Table 6. Non-Covering construct

Id	NCParent	NCChild	Conformance
nc1	L21	L18	p7

Table 7. Non-Onto construct

Id	NOParent	NOChild	Conformance
no1	L21	L24	h23

Table 8. Non-Strict construct

Id	NSParent	NSChild	Conformance
ns1	L21	L25	h25

3.2 Architecture

Schema evolution in this work is implemented as a web-based application to interact with a particular schema. Users perform schema evolution operations such as add/edit schema, dimension, level hierarchy, and path. Architecture of the application is shown in Fig. 2. It consists of a web interface as a front end/client side application and PHP as a back end/server side application that access MySQL.

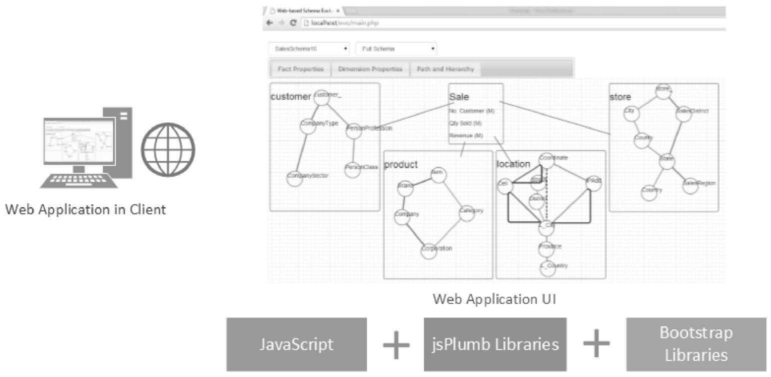

Fig. 3. Client-side application architecture

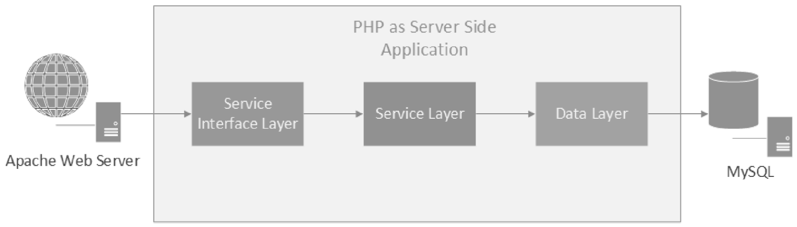

Fig. 4. Server-side application architecture

Client side application shows a user interface to allow users to view schema structure and perform schema evolution operations (Fig. 3). It is developed using JavaScript and API libraries of jspPlum and Bootstrap. Data exchange between client and server is done by using AJAX (Asynchronous JavaScript and XML) in JSON (JavaScript Object Notation) format. Then, backend processes in the server side validate the operations with associated constraints.

Fig. 4 shows the PHP/JSP server side application, which comprises a service interface, service and data layers. The service interface layer is responsible for routing client's requests to the service layer. It then calls a particular function in the service layer according to the request. The service layer contains all available methods the application has. The actual computation of the schema evolution operations happens in the data layer, which has access to the meta-schema database to query and modify the database. A return from the data layer can be data or status of the operations.

Data layer functions validate schema evolution operations with corresponding constraints. These evolution operations include multiple alternative, parallel dependent and parallel independent hierarchies [8] and core features and extended hierarchies (non-covering, non-onto and non-strict hierarchy), constraint functions [2].

3.3 Meta-schema Database

Meta-schema database stores metadata information of schemas in the form of relational tables. Schema evolution operations modify records in the tables. The meta-database consists of a schema construct table, which stores sets of dimension, fact and primary key constructs. The dimension construct has level, hierarchy and path constructs. The fact construct has measure constructs.

Conforming Dimensions

The application allows a user to create more than one schema to represent different data marts. The dimension, hierarchy, level, fact and measure constructs are available to all schemas, which means every change that occurs in the constructs are reflected to all existing schemas. The objective of this feature is to achieve conformity of all data marts, which emphasizes Kimball's data warehouse being a union of conformed data marts as mentioned in the Introductiuon

Schema Evolution Algorithm Example: add Multiple Alternative Hierarchy

Many functions have been developed to accommodate schema evolution operations such as adding, updating and deleting level, hierarchy, path, dimension, fact and measure in a schema. Below is an algorithm that adds a new multiple alternative (MA) hierarchy. The algorithm requires a dimension name, parent and child levels as input arguments. The parent level should be a level in an existing path. For the child level, the user must create a new attribute as the child level before creating a new MA hierarchy. The output is a new path added into the dimension. The new path created should converge to the same level in the existing path.

```
Input:   dimension name, child level, and parent level
Output: the updated dimension with a new path

Step 1: Check if the dimension is valid.
Step 2: Check if the dimension has at least one existing path.
Step 3: Check if the input child and parent levels are valid levels.
Step 4: Check if the parent level is part of an existing path.
Step 5: Check if the parent level is not the leaf level of the existing path.
Step 6: Create a new path by calling the addSimpleHierarchy function to create two
          hierarchies: one between the leaf and child level and another one between the
          child and parent level. If the parent level is not the root level, call
          addSimpleHierarchy to add the rest of the hierarchies of the existing path in-
          to the new path.
```

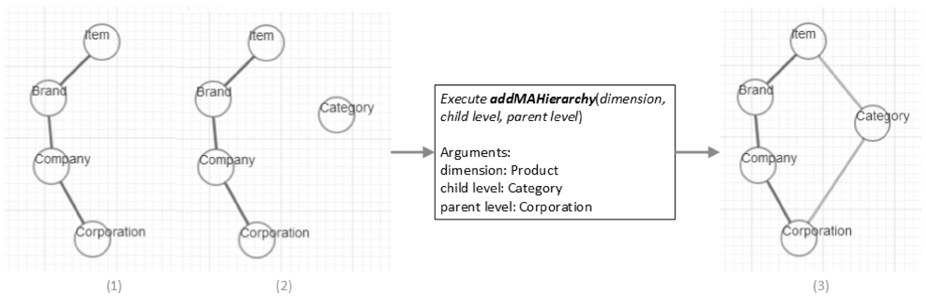

Fig. 5. Multiple Alternative hierarchy addition process (*addMAHierarchy)*

Fig. 5 illustrates the algorithm. There should be at least one existing path (Fig. 5(1)) and the new path must use the same level of the existing path as a parent level. In this case, the parent level is *Corporation*. For the child level, the user must add the new level, *Category*, before adding the new path (Fig. 5(2)). Then, the application executes the *addMAHierarchy* function, which requires three arguments namely, *Product* as the dimension name, *Category* as the child level and *Corporation* as the parent level. The function then creates a new path containing two simple hierarchies: 1) one between the leaf level of the existing path, which is *Item,* and the child level argument, which is *Category,* and 2) another one between the child level, *Category* and parent level arguments, *Corporation*. The *addMAHierarchy* function actually invokes the *addSimpleHierarchy* function to create these hierarchies and to add them into the path. As a result, Fig. 5(3) shows a newly created path as a Multiple Alternative hierarchy.

4 Conclusions

This paper presents a web-based application to implement data warehouse schema evolution, which allows the user to create, view and modify a schema of a multi-dimensional model based data warehouse. The application has objectives to allow the users to perform schema evolution operations in a user-friendly environment and to enforce schema correctness. The Internet browser-based system offers not only a platform independent environment but also broad user types beyond database experts. The application uses PHP functions to implement schema evolution operators to ensure schema correctness, and supports schema evolution over core features and extended hierarchies in multi-dimensional models. It also uses the MDD approach to not only implement such metadata constructs as dimension, level, hierarchy and path on a relational database, but also satisfies conforming dimensions among data mart schemas.

This work can be extended to include evolutions of more generalized dimensional hierarchies. Next level challenges would be extending this metadata-level conceptual schema evolution towards the physical level. The extension will work on modification of an underlying data warehouse schema and data in order to perform schema evolution operations.

References

1. Atzeni, P., Cappellari, P., Bernstein, P.A.: 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)
2. Banerjee, S., Davis, K.C.: Modeling data warehouse schema evolution over extended hierarchy semantics. Journal on Data Semantics XIII 5530, 72–96 (2009)
3. Bebel, B., Eder, J., Koncilia, C., Morzy, T., Wrembel, R.: Creation and management of versions in multiversion data warehouse. In: Symposium on Applied Computing, pp. 717–723. ACM, New York (2004)
4. Kimball, R., Ross, M.: The data warehouse toolkit, 3rd edn. John Wiley & Sons, Indianapolis (2013)
5. Malinowski, E., Zimányi, E.: OLAP hierarchies: A conceptual perspective. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 477–491. Springer, Heidelberg (2004)
6. Oueslati, W., Akaichi, J.: A survey on data warehouse evolution. International Journal of Database Management Systems 2(4), 11–24 (2010)
7. Talwar, K., Gosain, A.: Hierarchy classification for Data Warehouse: A Survey. Procedia Technology 6, 460–468 (2012)
8. Talwar, K., Gosain, A.: Implementing schema evolution in data warehouse through complex hierarchy semantics. International J of Scientific & Eng. Research. 3(7) (2012)
9. Gartner's Top 10 IT Trends of (2014), `http://www.news-sap.com/gartners-top-10-trends-2014/`