https://doi.org/10.48009/2_iis_2014_14-23
Issues in Information Systems
Volume 15, Issue II, pp. 14-23, 2014

DATA WAREHOUSE AND MASTER DATA MANAGEMENT EVOLUTION – A META-DATA-VAULT APPROACH

Danijela Subotic, Department of Informatics/University of Rijeka, dsubotic@inf.uniri.hr
Vladan Jovanovic, Georgia Southern University, vladan@georgiasouthern.edu
Patrizia Poscic, Department of Informatics/University of Rijeka, patrizia@inf.uniri.hr

ABSTRACT

The paper presents a: a) brief overview and analysis of existing approaches to the data warehouse (DW) evolution problem, and b) detailed description of the research idea for the DW evolution problem (primarily intended for structured data sources and realizations with relational database engines). We observe the DW evolution problem as a double issue - from the DW perspective, and from the master data management (MDM) perspective. The proposed general solution will include a Data Vault (DV) model based metadata repository that will integrate the DW data and metadata with the MDM data and metadata. This historicized metadata repository will manage schema versions and support schema changes. We believe this integration will: a) increase the quality of data in the Enterprise Data Warehouse (EDW), b) increase the quality of the metadata in the EDW, and c) increase the simplicity and efficiency of the DW and the MDM schema evolution.

Keywords: data warehouse evolution, master data management, data vault, metadata repository, schema evolution, schema versioning, view maintenance

INTRODUCTION

A data warehouse (DW) integrates current and historical data from a number of heterogeneous data sources into a central repository of data which is then used for business reporting, trend projections and quick data analysis. However, the DW environment is in a constant change – data sources often change their structure and content, and business needs and analytical requirements also change over time. These changes have to be propagated to and implemented into the DW, so the DW could accurately reflect the current state of the real world and provide for effective business analysis. Evolving the DW is a hard and resource consuming task and we observe it as a DW evolution problem. Although the database (DB) evolution problem is well researched, and some methods and solutions were borrowed and modified for the purpose of solving the DW evolution problem, the DW requirements today are increased in respect to the scope (multiple sources and multiple data types) and preservation of the history of changes. The DW needs to preserve the history of data and metadata changes, as well as the history of schema and scope changes, for a very long time period. Nowadays business and technology changes are constantly present, so it has become extremely important to find an appropriate solution to the DW evolution problem. Many researchers have studied this problem through three main approaches - schema evolution [4, 8, 11, 17, 21, 29], schema versioning [1, 5, 7, 9, 22, 23, 27] and view maintenance [2, 6, 10, 24, 25, 26]. In this paper we will briefly present related work and then proceed to describe our general research idea and approach for solving this problem. The paper is organized as follows – in section II a related work is briefly presented and analyzed, in section III our general research idea is described as a work in progress, and in section IV the conclusion and directions for our future work are presented.

RELATED WORK

As we mentioned in the Introduction, the DW schema evolution research has a very broad scope and the related work can be observed through the three main and above-mentioned approaches. In the schema evolution approach the schema can only have one version at a given time - the current version, where the data is stored. The current version is directly transformed into a new version, after the changes are made. This causes the loss of history (of data and schema changes), because the previous schema versions are not preserved. In order to avoid previously available structures and data becoming unavailable in the new version and enable the reconstruction of structures and data, schema versioning approach has been developed. In the schema versioning approach data is transferred from existing schema to the new schema. The changes are made to the new schema and the old schema versions are preserved. As this is basically the only difference between these two approaches, schema evolution approach is usually considered as the weaker case of schema versioning approach. Also, in both approaches the DW is usually

defined as a multidimensional schema with the fact and dimension tables and data cubes. Here authors usually study the schema changes in multidimensional DB, based on the various levels of updates (such as dimension, instance, level, fact, attribute, data cube, hierarchy, measure, constraint, quality or structure updates). Some authors propose new update operators [5, 9, 11, 17, 22], frameworks for multidimensional schema evolution and versioning [4, 27], dimension and data cube description and visualization tools and prototypes [29], and transformation, migration and mapping models, mechanisms, systems and prototypes [1, 7, 8, 21, 23]. We noticed that the problem of change propagation in data mart (DM) is quite well researched and many good and often complementary solutions were proposed. However, the process of schema evolution and versioning is still demanding in terms of invested time and resources (particularly at the present time due to the increase in the number of changes in the data sources and user requirements). In the context of schema evolution and versioning we can say there is room for improvement with regard to a still slow and expensive processes of data transformation and migration between different schema versions, loss of information during those same processes (the problem of preservation of schema consistency and data integrity still exists), inefficient query and application rewriting and adapting to the new schema version, inefficient and error prone cross-querying the versions, a lack of effective integration, organization and management of metadata, a lack of defined mechanism for monitoring the data source model evolution, a lack of support for the model relativism, and a lack of support for the data security evolution. In the view maintenance approach a DW is defined as a set of materialized views over data sources. View maintenance approach can be categorized further into two approaches: view adaptation (metadata with the latest structural information is added to materialized view in order to adapt to changes) and view synchronization (rewriting the view after the changes in data sources). Here authors usually study and propose the algorithms for (basic and incremental) view maintenance and lineage tracing [6, 10], formal frameworks, models and update operators for view maintenance [26], mechanisms for view versions [2], mechanisms, systems and prototypes for view rewriting [25] and mechanisms for obtaining and evaluating quality of data during the schema evolution [24]. Although many interesting solutions are proposed, manual rewriting of the view is still a common practice. Also, to the best of our knowledge, the problems of network saturation, anomalies and inconsistent changes in the views are still present, the proposed approaches are still limited in terms of efficiency and performance and ETL processes are completely ignored in this approach.

RESEARCH METHODOLOGY

Problem

The problem of the DW evolution is the propagation of (and adaptation to) constant changes in its environment (primarily including changes in data sources and business requirements). As we already mentioned, the DW needs to preserve the history of data and metadata changes, as well as the history of schema and scope changes, for a very long time period. Literature analysis shows us that many of the important issues are still not quite successfully resolved. We will describe our general research idea for solving some of the above mentioned issues.

Research idea

We approach the DW evolution problem from two perspectives – the usual data warehouse perspective, and the master data management perspective. Master Data Management (MDM) has been traditionally used as a physically independent database of master and reference data and definitions for operating systems [3]. MDM represents the set of policies, governance, standards, processes and tools that define and manage the master and reference data of a business organization to provide a single point of reference. Master data are the key business entities and their descriptive attributes (e.g., the buyer has the name, address, date of birth, etc.). They represent a unique source of essential business data and are used by multiple (ideally all) systems, applications, and business processes of the organization. Reference data are used for the validation of other data and they define the set of allowed values that the other data fields may use. In the case of MDM, as in a DW case, there is the problem of schema evolution after changes in the data sources or user requirements – so we will consider it a double issue or a dual problem. As such we will aim to develop a dual solution in which the DW and the MDM evolution problem will no longer be managed separately, but together (the MDM integrated with the DW). From the DW perspective every fact that is associated with dimensions is observed and star schemas are standard representation used for visualization. From the MDM perspective every master entity (dimension) that is associated with events (facts) is observed and an inverse star-like schema can be used for visualization [18].

Figure 1 shows an architecture diagram for our future dual solution. It includes a Data Vault (DV) based modeling approach [15, 19] for the data repository (RDV + BDV) and the metadata repository (MDV). We use Data Vault based modeling approach because of its many advantages, and also because we believe that the relational model alone, as a logical model, is not convenient for effective and simple DW schema evolution support as it does not separate identity from the properties (attributes and relationships). The Data Vault (DV) is a data modeling method that supports design of data warehouses for long-term storage of historical data collected from various data sources. The DV method is already based on the assumption that the DW environment is in constant change and it highlights the need for tracking the origin of data contained in the database, through empirically defined set of metadata. This enables tracking the value back to the source and tracking the history of changes. Also, according to the DV method, there is no difference between good and bad data - all the data is stored at all times, regardless of whether they are adaptable to business rules, thus avoiding the loss of information. The structural data are explicitly separated from descriptive attributes, regardless of whether they come from the same source. This makes the model flexible to changes in business environment, and allows for a gap analysis and trend projections (patterns can be mined from a single sheet of business keys and links can have dynamic adaptability). Furthermore, any change is implemented in the model as an independent extension of the existing model, which means that the changes do not affect current applications. This also means that all versions of the application can be based on the same, developing DB. All versions of the model are a subset of the DV model. Finally, the DV method enables fast parallel loading which reduces the costs of time and other resources. In our proposed dual solution architecture (Figure 1), the data repository (RDV + BDV) will preserve raw and master data as well as history of data and data changes. Metadata repository (MDV) will preserve metadata, as well as history of metadata and metadata changes. Also, MDV will integrate the DW metadata with the MDM metadata in a common model to serve as an extension of a generic DBMS catalog. This way, the problem of the DW and the MDM schema evolution will be addressed at the general level and a permanent general solution situated on a higher (meta) level will be developed. The end result will be a flexible, modular solution which will be able to track and manage changes in both data and metadata, as well as their schemas. We will now explain the proposed architecture from Figure 1 in more detail. The proposed architecture consists of four parts: a) data sources, b) enterprise DW, c) reporting DW and d) user analysis.

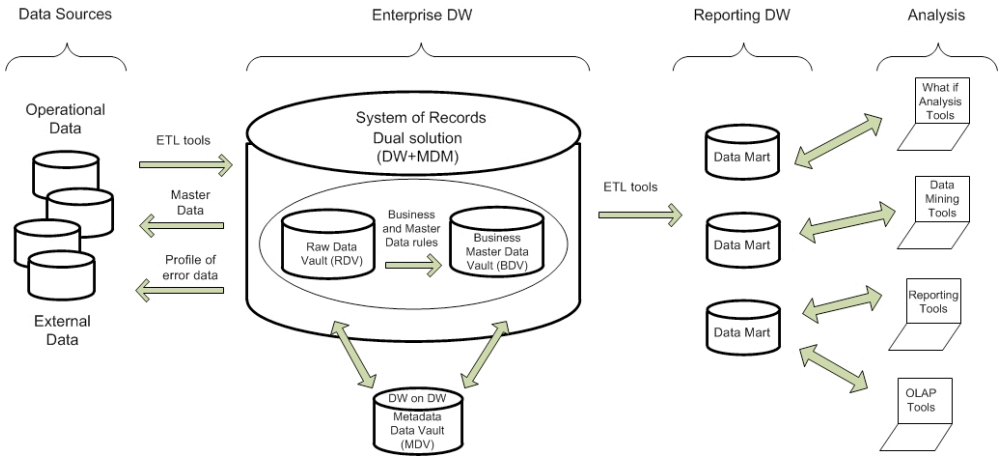

Figure 1. Architecture Diagram

a) Data Sources

The DW today includes many heterogeneous data sources. In the literature and practice, data sources are usually distinguished by their place of origin and maintenance. Internal (or operating) sources contain data that are defined, related and maintained in various parts of the business organization (e.g. orders from sales, inventory status from the warehouse, potential customers from marketing, etc.). External sources contain information that is not collected by a business organization, rather a source outside the organization (e.g. various proprietary databases, address and phone directories, etc.). Accordingly, we make a distinction between internal and external reference data (which is obtained from internal or external data sources).

b) Enterprise DW

Enterprise DW (EDW) includes the raw data vault (RDV) and business master data vault (BDV), which are integrated via metadata repository data vault (MDV).

With the help of ETL tools and processes data is extracted from data sources and loaded into the RDV. The RDV contains actual (unchanged) copies of the originals from the data sources. Once the data is entered into the RDV, it is no longer deleted (all future changes are implemented through additions-only). This means that copies of the originals are kept permanently in RDV. In this way the loss of information is avoided and a solid basis for the audit process is provided. We can say that the RDV is a component of the EDW which is data source-oriented.

The BDV is a component of the EDW which is report-oriented. It is created by upgrading the RDV with the application of standardized master data and business rules, for the purpose of business integration. The RDV and the BDV are shown as separate systems in a logical representation in Figure 1. However it is possible to physically implement them individually or as a single system. We will physically implement them as a single database (i.e. one DV model) where the RDV and the BDV structures are connected via the same-as links.

EDW consists of a single DV model, which is partially oriented towards the data source side (RDV), and partially oriented towards the MDM and reporting side (BDV). Also, because they are now physically separated, we can distinguish reversible (light) and irreversible (heavy) transformations [16]. Reversible transformations are used for loading data from a data source into the RDV, and it is possible to reverse their effects, in order to obtain the system-of records. They allow RDV to reach the exact copies of the original from the data source. Irreversible transformations are mainly based on the business and master rules and are usually irreversible. In this case, both the transformations and the original data must be preserved in order to trace exits back to the source and reconstruct them if necessary. That is why it is necessary to preserve the RDV part of EDW. The irreversible transformations are moved downstream - after RDV, towards BDV and reporting DW (they are loading materialized DMs). This is the key idea for getting an integrated EDW system of records [16, 19] and a basis for the data governance.

Finally, MDV is the metadata repository which serves to integrate the RDV and BDV in the EDW system of records. We can say that the MDV represents the DW on the DW. MDV logical model is based on the DV model. The aim of our research is to define and formalize the MDV model for the integration of dual solution EDW (DW and MDM).

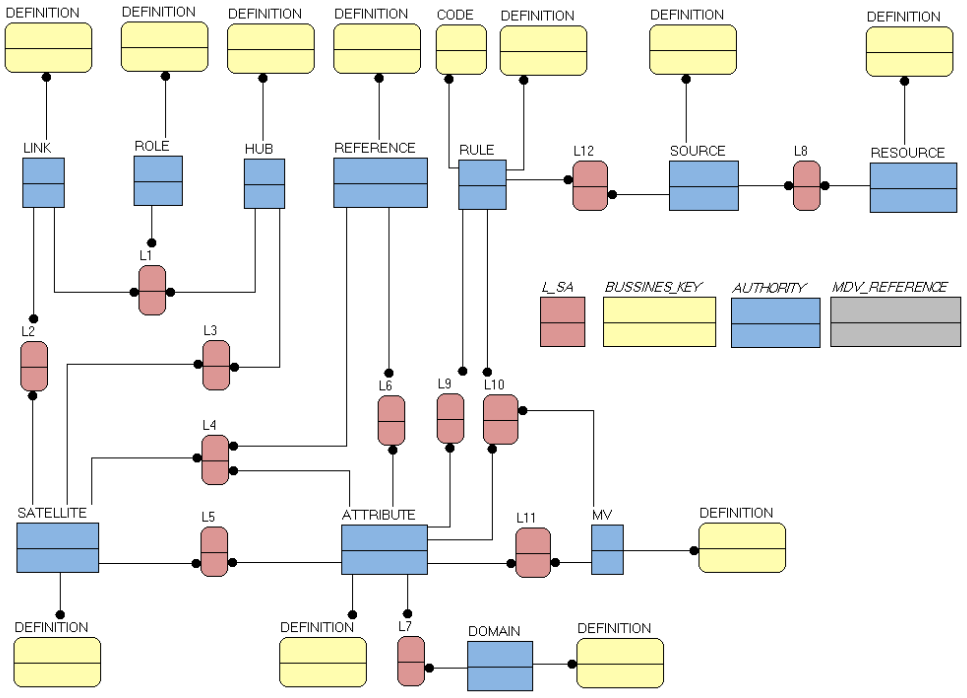

Figure 2. Simplified Working Version of the Metadata Data Vault (MDV) Model

Figure 2 shows a simplified working version of the MDV model. We see that the MDV keeps historicized hubs, links, satellites, attributes, domains and reference tables as a hub in a DV model (shown in blue). All hubs in MDV

have a corresponding DEFINITION satellite attached to them (shown in yellow). Standard details of DV meta-level attributes (loadDTS, loadEndDTS, recSource, busKey, etc.) are not shown (for the purpose of a readability of the model) but apply to all DV constructs at the MDV level. Links between entities in the MDV (shown in red) are also simplified for the purpose of readability of the model. L_SA are same-as links connected to other hubs in the model and they serve to define master HUBS, LINKS, SATELLITES, ATTRIBUTES, SOURCES, RULES, etc. BUSINESS_KEY are satellites connected to every hub in the model (business key is placed into a separate satellite). We also plan to resolve (at the meta-level) the transformation from a source into the RDV (via the extraction rules; SOURCE-L12-RULE), the transformation from the RDV into the BDV (via the business and master rules; RULE-L9-ATTRIBUTE) and finally, the transformation from the integrated RDV/BDV into the DM (also via the business rules and materialized views; RULE--ATTRIBUTE-MV- L10). Furthermore on a meta-level, the security aspect will be resolved, i.e. the user's access rights will be historicized and managed. Considering that, SOURCE is a hub representing source systems and RULE is a hub representing business and master rules (with corresponding CODE satellite). AUTHORITY is a hub linked to other hubs in the MDV, for security aspect and user access rights. A MV (materialized view) is a transformation from the integrated RDV/BDV into the DM and is shown as a hub. Metadata reference tables (for example, internal metadata categorization such as metadata entities type codes and descriptions and all additional views) will be treated as MDV_REFERENCE (shown in gray). MDV will manage master sources, rules and other MDV entities, which is shown as a same-as link L_SA on every hub in the model.
The proposed architecture permanently stores historical and master data into the EDW, which is why there is a return link to the data sources. This not only solves the problem of the DW evolution, but the MDM evolution as well, at the same (integrated) level. Also the data definitions, which are usually stored in the MDM, are now stored in the MDV together with the rules of transformations (specifications for the ETL code), and here they are treated as regular data. In this way historicized metadata repository (MDV) manages schema versions and supports schema changes. However, we would stress that the MDV model shown in Figure 2 is a fairly preliminary, a true working (draft) version for now, because the research is in progress. We are still working on a source, materialized view, transformation and security aspects, which are not shown in detail in a MDV model in Figure 2. For this reason, the formal definition and description for each component of the MDV model is not presented.

c) Reporting DW
Reporting DW (RDW) consists of a derivative (summarized, aggregated and computed) data stored in materialized or virtual DM. User has a direct access to the data stored in RDW, for the purpose of analysis and reporting.

d) User analysis
This part of the architecture is the user side of the system, where the tools for analysis and reporting are located. With the help of these tools the user is directly accessing the RDW.

Research validation approach

Our research centers upon the following question: Can proposed DW and the MDM integration through the MDV metadata repository demonstrably serve as a permanent general solution able to effectively track and manage changes in both data and metadata (including their schemas?)
Once proposed MDV model and set of change cases as requirements are completed and fully formalized we intend to develop logical arguments for validating principal claims of the proposed approach. To empirically validate proposed approach an early system prototype will be developed, as a full proof of concept implementation, with a performance based experimental benchmark test based on a completed comprehensive case study model. As our work is still evolving all this will not be presented here, but in the meantime, and to serve us as a stepping stone in exploring problems and refining details for the solution a case study will be experimented with.

Research requirements

A key component in our dual solution is obviously the MDV, which can also be observed as a DW on the DW. MDV will serve to integrate raw and persistent DW with the business aligned MDM in order to obtain one consolidated EDW system of records. In the context of DW evolution, starting requirements for our dual solution are (we expect MDV will be able to): 1) track the origin of data, 2) track the history of changes (of data and metadata), 3) provide a mechanism for monitoring data source model evolution, 4) provide a mechanism for monitoring user requirements evolution, 5) support the data security evolution, 6) avoid loss of information, 7) enable faster and less

expensive migration and transformation of data, 8) support effortless cross-version queries, 9) enable trend projections, and 10) provide effective integration, organization and management of metadata.
In order to better understand the system requirements and effects of the proposed approach we will explore basic schema evolutions change cases, see Table 1, later on.
For the analysis of requirements, we will use a business case which deals with a DW for the outdoor and adventure equipment sales company (high rate of change in marketing campaign budgets and product price lists and categorization). All data model examples are made in Erwin 9.5 and are based on IDEF1X (Figures 2 to 4).

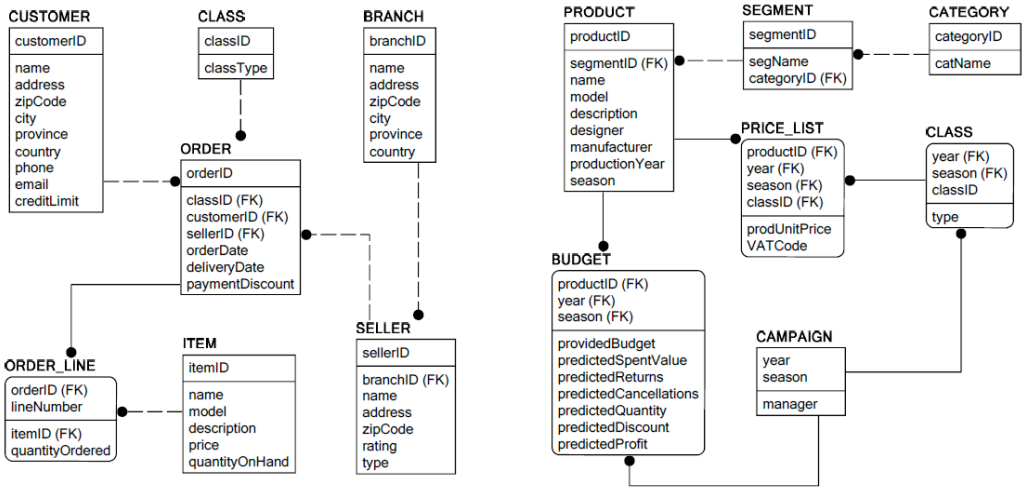

Figure 3. Data models for source databases

Figure 3 shows two simplified logical entity-relationship (ER) models for two data sources – Sales DB and Marketing DB. The Sales DB represents a sale of ITEM to CUSTOMER where SELLER places an ORDER for multiple items (one per ORDER_LINE). Every seller belongs to a branch, and every order has its class (which affects the amount of payment discount). The Marketing DB serves to manage marketing CAMPAIGNs. Every campaign has its own PRICE_LIST, which is created considering the CLASS of campaign and the PRODUCT. Products are grouped into SEGMENTs, and segments are grouped into CATEGORYs. Also, campaign has marketing BUDGET for every product.
Figure 4 shows logical DV model for our dual solution (RDV+BDV). Each hub, link and reference table have load date time stamp (loadDTS) and record source (recSRC) attribute, and each satellite has load end date time stamp (loadEndDTS) and record source (recSRC) attribute. We will just not show those attributes in the Figure 4, for the purpose of the readability of the model. Also for simplicity business keys were used instead of recommended sequence number keys. BDV model is shown as a second layer, an extension of the RDV model (by applying business and master rules). The RDV logical model is represented in darker shade of blue (hubs), red (links) and yellow (satellites) and the BDV logical model is represented in lighter shades of blue, red and yellow, respectively. Reference tables are shown in light gray and are managed in the BDV layer.
Hubs in the RDV (source entities) are through MDM links connected to the appropriate master hub in the BDV (master/target entities). This MDM link maps an entity to a standard (master) entity at any given point in time. Every master hub has at least one satellite (with the descriptive attributes) and one same-as MDM link. Same-as MDM links represent hierarchical (parent-child) or different business relationships between the master entities and serve to recognize and eliminate duplicates (duplicate record becomes a child to the master record parent). They map an entity to other entities within a single table, at any given point in time. For example, customer hub in RDV (H_CUST) is cross-referenced to customer master hub in BDV (H_CUST_MASTER) through MDM link (L_CUST_MDM). H_CUST_MASTER contains the "golden copy" of the customer's data and has a same-as MDM link (SAL_CUST_MDM) that maps a record to other records within a H_CUST_MASTER hub.

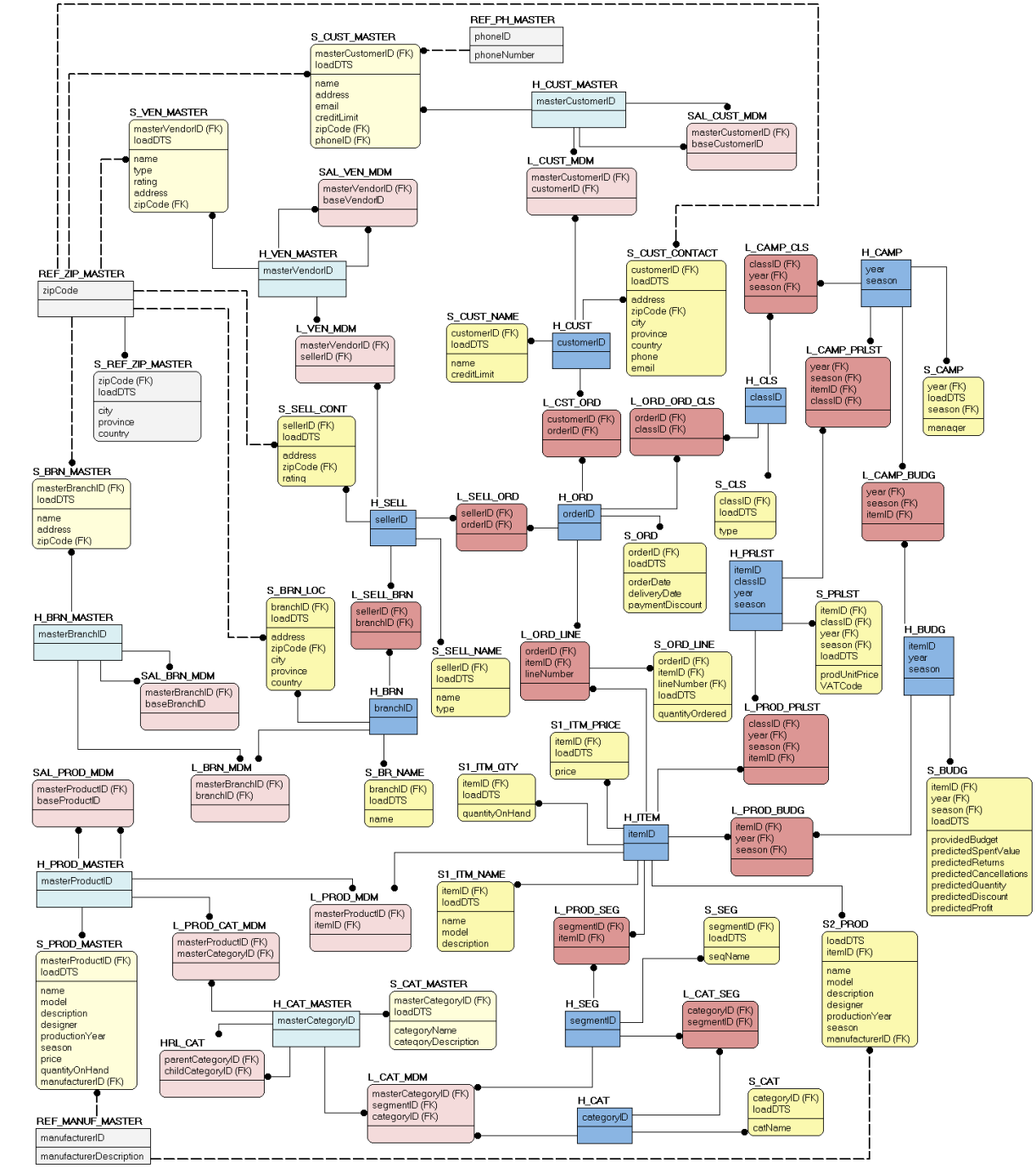

Figure 4. Data model for our dual solution (RDV+BDV)

Some of the typical DW evolution scenarios are changes in the existing data source schemas, additions of a new data sources, changes in the user requirements (business rules), changes in types of analytical systems or data sources, and changes in user access control (security). In this paper we will observe only the effects of changes in the existing data source schemas on the RDV, BDV and MDV (the DM is out of the scope of this paper). In Table 1 we observe following changes in the existing data source schemas and their effects on the RDV, BDV and MDV: Addition of the new relation, Modification of an existing relation, Deletion of the existing relation, Addition of a new attribute to

an existing relation, Modification of an existing attribute in the relation, Deletion of an existing attribute from the relation, Addition of the new relationship between existing relations and Deletion of an existing relationship.
Similarly to the effects described in the Table 1, addition of a new data source implies the creation of new hubs, links, satellites and reference tables in the RDV and the use of new and existing business rules in the BDV, as well as implementing appropriate changes of metadata in the MDV. With additions of a new data sources we can apply the same changes we applied in the previous case (changes in the existing data source schemas), with some additional changes in the MDV: a) a new record is added into SOURCE hub and SOURCE_DEFINITION satellite, b) new records are added into RULE hub and RULE_DEFINITION and CODE satellites, c) new records are added into RULE-SOURCE and RULE-ATTRIBUTE links, and d) possibly new records are added into SOURCE and RULE same-as links (for defining master sources and rules). Changes in the user requirements (business rules – which primarily affect the BDV and DMs), changes in types of analytical systems or data sources, changes in materialized views, and changes in user access control (security) are very important but are still subject of evolving research.

Table 1. Effect of changes in the existing data source schemas on the RDV, BDV and MDV

Source schema change	Example	Effect on RDV	Effect on BDV	Effect on MDV
Addition of a new relation	SUBCATEGORY in Marketing DB	- addition of new hubs, links and satellites (simple extension of the model) - addition of a validity satellite for the SEGMENT-CATEGORY link	- addition of a new link (L_CAT_MDM) which connects category master hub in BDV with segment, subcategory and category hubs in RDV	- addition of new records in the LINK, HUB, SATELLITE and ATTRIBUTE hubs (for SUBCATEGORY) - addition of a new record in a RULE hub and RULE_DEF and CODE satellites
Modification of an existing relation	SELLER into VENDOR in Sales DB	- no action necessary	- no action necessary	- addition of a new record into HUB_DEF satellite - addition of a new record into RULE_DEF and CODE satellite
Deletion of an existing relation	CATEGORY from Marketing DB	- no action, no deletions, updates only - addition of a new validity satellite on a relation hub and all its links	- no action, no deletions, updates only - addition of a new validity satellite on a master hub	- addition of a new record into HUB's validity satellite - addition of a new record into RULE_DEF and CODE satellite
Addition of a new attribute	COLOR into ITEM in Sales DB	- addition of a new satellite - possibly consolidate old and new satellite	- addition of a new satellite - possibly consolidate old and new satellite	- addition of a new record into ATTRIBUTE hub - addition of a new record into ATTR-DOMAIN, ATTR-SATELLITE, ATTR-RULE and ATTR-RULE-MV link - addition of a new record into ATT_DEF satellite
Modification of an existing attribute	PRICE domain in ITEM into DECIMAL or PRICE into UNIT_PRICE in Sales DB	- addition of a new ITEM hub satellite	- addition of a new PRODUCT_MASTER satellite	- addition of a new record into ATTR-DOMAIN link - addition of a new record into SATELLITE and ATTRIBUTE hub - addition of a new record into RULE_DEF and CODE satellite
Deletion of an existing attribute	DESCRIPTION in ITEM in Sales DB	- no action, no deletions, updates only - addition of a new validity satellite on a ITEM hub	- no action, no deletions, updates only - addition of a new validity satellite on a PRODUCT_MASTER hub	- addition of a new record into ATT_DEF with validity data - addition of a new record into RULE_DEF and CODE satellite
Addition of a new relationship	BRANCH-CUSTOMER	- addition of a new link (simple extension of the model)	- addition of a new link (simple extension of the model)	- addition of new records in the LINK hub, LNK_DEF satellite and LINK-HUB-ROLE link - addition of a new record into RULE_DEF and CODE satellite
Deletion of an existing relationship	BRANCH-CUSTOMER	- addition of a new validity satellite on a BRANCH-CUSTOMER link	- addition of a new validity satellite on a BRANCH-CUSTOMER link	- addition of a new validity satellite on a LINK hub - addition of a new record into RULE_DEF and CODE satellite

CONCLUSIONS

The DW environment is in a constant change (nowadays more than ever), but the DW evolution process is still quite complex, error prone and requires a lot of time and resources. In this paper we conducted a brief analysis of the previous DW schema evolution research and presented a detailed description of the proposed idea for the further research. Through the analysis of the related literature, we noticed the lack of broader, general approaches for solving more of the DW schema evolution problems together, as well as the lack of metadata repository standardization for the support of the DW schema evolution. Considering that, we will try to tackle the DW schema evolution problems on a more general, meta-oriented level. We will approach the DW evolution problem from the usual data warehouse perspective, but also from the master data management (MDM) perspective. The problem of schema evolution after the changes in the data sources or user requirements is present in both of these systems and we believe they should be dealt with together, on the same level. The research idea described in the paper includes creation of a dual solution - an integrated DW and MDM data repository, which is primarily intended for structured data sources and realizations with relational database engines. The DW and the MDM will be integrated through the standardized and historicized metadata repository (MDV), that will be based on a Data Vault modeling approach. We believe standardized and historicized metadata repository is the key of solving the DW and the MDM schema evolution problem, together with the usage of the Data Vault (DV) modeling approach. The DV modeling approach, because it highlights the need for tracking the origin of data and history of changes, is more convenient to provide simple and effective support for the historical DW schema evolution than the relational modeling approach. A MDV can in this context be observed as a DW on a DW, and it will serve to integrate raw data oriented DW with a business and master data oriented MDM. Our dual solution will aim to solve many of the DW evolution problems mentioned in the paper (such as a slow and costly migration and transformation, loss of information, a lack of a mechanism for monitoring data source model evolution, user requirements evolution and data security evolution, a lack of effective integration, organization and management of metadata, etc.) together, at a higher, meta level.

By integrating the DW with the MDM through the MDV, our proposed dual solution would support simpler and faster DW/MDM evolution and it would extend the lifetime of the DW, by ensuring higher level of users' faith in the accuracy and validity of data used for reporting and analysis, as well as support for agile development (due to the modularity of the DV model). It would be able to deal with data quality issues across all master data types, applications, and areas and will enable business organizations to maximize the value they can gain from their master data. Also, it would contain both a "single version of the fact" [19] and a "single version of the truth" [13] and it could be used as a complete system of records [13, 16, 19] with the support for the data security evolution, data audit and data governance (it would provide a mechanism for monitoring data and metadata quality, as well as a mechanism for monitoring the DW and the MDM evolution). Among directions of ongoing and future research is a further gathering and defining of the solution requirements, standardization and formalization of a data vault based metadata repository (including source, materialized view, transformation and security metadata) by incorporating some general ideas from [12, 14, 20, 28] and incremental development of a implementation prototype for testing the research hypotheses and solution.

ACKNOWLEDGEMENTS

This paper is based upon work supported by the University of Rijeka under project titled "Metode i modeli za dizajn i evoluciju skladišta podataka".

REFERENCES

1. Bebel, B., Eder, J., Koncilia, C. , Morzy, T., & Wrembel, R. (2004). Creation and Management of Versions in Multiversion Data Warehouse. In 19th ACM Symposium on Applied Computing (SAC 04), Nicosia, Cyprus, 717–723.
2. Bellahsene, Z. (2002). Schema Evolution in Data Warehouses. Knowledge and Information Systems, 4(3):283–304.
3. Berson, A., & Dubbov, L., (2011). Master data management and data governance, 2nd edition, McGraw Hill.
4. Blaschka, M., Sapia, C., & Hofling, G. (1999). On Schema Evolution in Multi-dimensional Databases. In 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK 99), Florence, Italy, vol. 1676 of LNCS Springer,153–164.
5. Body, M. , Miquel, M., Bedard, Y., & Tchounikine, A. (2002). A Multidimensional and Multiversion Structure for OLAP Applications. In 5th ACM International Workshop on Data Warehousing and OLAP (DOLAP 02), McLean, Virginia, USA, 1–6.

6. Cui, Y., & Widom, J. (2000). Practical Lineage Tracing in Data Warehouses. In Proceedings of the 16th International Conference on Data Engineering (ICDE'00), San Diego, California.
7. Eder, J., & Koncilla, C. (2000). Evolution of Dimension Data in Temporal Data Warehouses. Technical Report.
8. Fan, H., & Poulovassilis, A. (2004). Schema Evolution in Data Warehousing Environments – a schema transformation-based approach. In Proceedings of Conceptual Modeling - ER, 23rd International Conference on Conceptual Modeling, Shanghai, China.
9. Golfarelli, M., Lechtenbörger, J., Rizzi, S., & Vossen, G. (2004). 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 Springer, Heidelberg, vol. 3289, 415–428.
10. Gupta, A., & Mumick, I. (1995). Maintenance of Materialized Views: Problems, Techniques, and Applications. Data Engineering Bulletin.
11. Hurtado, C. A., Mendelzon, A. O., & Vaisman, A. A. (1999). Maintaining Data Cubes under Dimension Updates. In Proceedings of the 15th International Conference on Data Engineering (ICDE), Sydney, Australia, IEEE Computer Society, 346–355.
12. Inmon, W., Zachman, J., & Geiger, J., (1997). Data stores, data warehousing, and the Zachman's framweork, McGraw Hill.
13. Inmon, W.H., Strauss, D., & Neushloss, G. (2008). DW 2.0: The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann Publishers, Burlington, USA.
14. Jiang, B., (2011). Constructing data warehouse with metadata-driven generic operators, and more, DBJ Publishing.
15. Jovanović, V., & Bojičić, I. (2012). Conceptual Data Vault Model. In Proceedings of the Southern Association for Information Systems Conference, Atlanta, USA.
16. Jovanović, V., Bojičić, I., Knowles, C., & Pavlic, M. (2012). Persistent Staging Area Models For Data Warehouses. Issues in Information Systems, vol.13, iss. 1, 121-132.
17. Kaas, C.E., Pedersen, T.B., & Rasmussen, B.D. (2004). Schema Evolution for Stars and Snowflakes. In Proceedings of the International Conference on Enterprise Information Systems (ICEIS 2004), Portugal.
18. Kimball, R., & Ross, M., (2013). The data warehouse toolkit, 3rd edition, John Wiley.
19. Linstedt, D. (2011). SuperCharge Your Data Watehouse: Invaluable Data Modeling Rules to Implement Your Data Vault. CreateSpace Independent Publishing Platform, USA.
20. Marco, D., & Jennings, M., (2004). Universal meta data models, John Willey.
21. Marotta, A., & Ruggia, R. (2002) Data Warehouse Design: A schema-transformation approach. 22nd International Conference of the Chilean Computer Science Society (SCCC), Copiapo, Chile.
22. Morzy, T., & Wrembel, R. (2004). On Querying Versions of Multiversion Data Warehouse. In Proceedings of the International Workshop on Data Warehousing and OLAP, DOLAP‟04, Washington, USA.
23. Papastefanatos, G., Vassiliadis, P., Simitsis, A., & Vassiliou, Y. (2007). What-if Analysis for Data Warehouse Evolution. In Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery, Regensburg, Germany.
24. Quix. (2004). Repository Support for Data Warehouse Evolution. In Proceedings of the International Workshop DMDW, Heidelberg, Germany.
25. Rundensteiner, E. A. , Koeller, A., Zhang, X., Lee, A.J., & Nica, A. (1999). Evolvable View Environment EVE: A Data Warehouse System Handling Schema and Data Changes of Distributed Sources. In Proceedings of the International Database Engineering and Application Symposium (IDEAS'99), Montreal, Canada.
26. Rundensteiner, E. A., Koeller, A., & Zhang, X. (2000). Maintaining Data Warehouses Over Changing Information Sources. In Communications of the ACM, vol.43, 57-62, New York, USA.
27. Solodovnikova, D. (2007). Data Warehouse Evolution Framework. In Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, Moscow, Russia.
28. Tannenbaum, A., (2002). Metadata solution, Addison Wesley.
29. Vaisman, A.A., Mendelzon, A.O., Ruaro, W., & Cymerman, S.G. (2002). Supporting Dimension Updates in an OLAP Server. In Proceedings of the CAISE02 Conference, Canada.