Reinventing the Wheel?! Why Harmonization and Reuse Fail in Complex Data Warehouse Environments and a Proposed Solution to the Problem

Torsten Priebe
Teradata GmbH
Storchengasse 1
A-1150 Vienna, Austria
torsten.priebe@teradata.com

Andreas Reisser
University of Regensburg
Universitätsstrasse 31
D-93053 Regensburg, Germany
andreas.reisser@wiwi.uni-regensburg.de

Duong Thi Anh Hoang
Vienna University of Technology
Favoritenstrasse 9-11/188
A-1040 Vienna, Austria
htaduong@ifs.tuwien.ac.at

ABSTRACT

Enterprise or group data warehouses are often introduced in complex multi-national organizations in order to foster harmonization, integrate heterogeneous source systems and hide the heterogeneity from analytical systems. Industry reference data warehouse logical data models such as Teradata's FS-LDM or IBM's BDW are promoted as accelerators for the development of such large data warehouses. However, this paper shows that logical data models alone are not sufficient to ensure reusability in those environments. In order to provide a solid basis for standardization, the logical data model needs to be accompanied by a semantic business information model used as an anchor point for the mappings and for communication with business users. Such a model allows a model-driven approach for specification of the data transformations, which usually accounts for at least half the total effort of large data warehouse projects. The paper presents an approach building upon the Teradata Business Data Element (BDE) concept giving practical examples from project experience in the financial services industry. A research prototype is presented, utilizing Semantic Web technologies such as the Web Ontology Language (OWL) to facilitate the traceability of data requirements, business terms and physical data elements in the different layers of a complex data warehouse architecture.

Keywords

Data Warehouse, Harmonization, Reuse, Traceability, Semantics, Data Modeling, Model-driven Engineering

1. INTRODUCTION

Large and complex multi-national organizations are increasingly aiming for enterprise or even group-wide data warehouse approaches to support their analytical needs efficiently and to facilitate harmonization and integration. Besides the advantages that true enterprise data warehouses (EDWHs) can deliver such as lower total-cost-of-ownership or a holistic view of the business, there is a high effort associated with the development and maintenance of a common data model and the mappings to source systems, analytical applications, semantic BI tool layers, etc., especially if the operational IT landscape is diverse and heterogeneous.

Real-world EDWHs usually contain thousands of data elements, which leads to several thousands of data mapping rules. In many cases these specifications are held in Excel spreadsheets or small databases, which are typically very hard to maintain. Often, specification and mapping account for about half the total effort and budget of large DWH projects. There are industry reference logical DWH data models, which accelerate and assure quality of the DWH evolution over time. However, as we will show in this paper, such logical data models are not sufficient to ensure traceability and enable harmonization and reuse in such complex environments. Experience shows, that standardized group or enterprise data warehouse projects in heterogeneous multi-national organizations often do not show the expected business value or leverage the expected synergies in the foreseen timeframe.

Furthermore, agile and self-service business intelligence approaches aim at accellerating analyses and thus reducing time-to-market by enabling business users without being dependent on big and sluggish IT projects. However, research by The Data Warehouse Institute (TDWI)[1] shows that 80% of business users are incapable of creating their own reports. One major barrier that keeps them from independently performing analyses is a lack of understanding of the logical and physical data models in place. Highly normalized EDWH data models are often hard to comprehend and query for business users and do not cover all semantics (in business terminology) needed to describe which data is available where and how it can be accessed.

It should be noted that this paper presents work in progress. Data warehouses are integrating information about business concepts from various sources and sharing it between dif-

10^{th} International Conference on Wirtschaftsinformatik,
$16^{th} - 18^{th}$ February 2011, Zurich, Switzerland

[1] http://www.tdwi.org

ferent target user groups and systems. In order to enable harmonization, it is necessary to map the different representations, of which one is the DWH itself (with its logical and physical data model), to a central point of reference which is the pure business concept. The main contribution of this paper is to examine the usefulness of a such a semantic business information model on top of logical DWH data models and to define its role within the data warehouse engineering process. It also shows how the Teradata Business Data Element (BDE) concept and Semantic Web Technologies can provide a basis for supporting this approach. However, Semantic Web technologies are currently really just used as a framework to implement a traceability prototype, based on the work presented in [13]. Our current follow-up research focuses on defining in detail, how this model should be built (e.g. using OWL modeling constructs instead of representing BDEs as OWL individuals) in the context of a model-driven data warehouse engineering approach.

The rest of this paper is organized as follows: Section 2 gives a thorough problem definition backed by some real-world examples from the authors' project experience. Section 3 discusses related work, both scientific and by DWH software vendors. In section 4 we outline our proposed solution, which we base on the Business Data Element (BDE) concept recently defined by Teradata and our preliminary research on data tracing with Semantic Web technologies [13]. We will give a thorough coverage of our future work and a conclusion of the paper in section 5.

2. PROBLEM DEFINITION

2.1 Harmonization and Reuse in Complex DWH Environments

Enterprise data warehouses are often introduced as a means of integrating heterogeneous source system landscapes and to hide this heterogeneity from analytical systems. The DWH data model is used to decouple the analytical world from the various different source systems with their different data models. This approach can be found particularly in multi-national organizations, which went through mergers and acquisitions and whose operational IT landscapes cannot be harmonized that easily. For example, the authors have worked with different groups in the financial services industry which have grown rapidly in emerging markets such as Central and Eastern Europe.

Figure 1 shows different architecture schemes for such enterprise or group data warehouses. Scheme (a) represents the approach of defining a standardized local DWH in order to reuse analytical applications and transformations from the DWH to application data marts. Scheme (b) adds to this a group layer in form of a central enterprise data warehouse (EDWH) for group-level reporting and applications. Finally, scheme (c) depicts a purely central approach without local DWHs (or with an unstandardized local DWH landscape, the point is that the reuse is accomplished through the central EDWH). In all three cases there is a standardized DWH data model, which is usually built with the following requirements in mind[2]:

- It should be independent of particular source systems, which may change over time or even be replaced due to harmonization and standardization initiatives.
- The data model should enable the standardization of data across the different entities (with different source systems, and to some degree even different business processes).
- It should allow business users to easily understand, which data is included in the DWH and where it can be found. Data requirements should be easily traceable to their physical location in the DWH and back to the source systems they are originating from.
- As changes in the physical data model impact a large number of data load jobs (consider, say, 10 different source systems), it should be able to cope with changing data requirements without having to change the the physical database schema as frequently.

These requirements are often addressed by basing the data model on an industry reference model, such as the Teradata Financial Services Logical Data Model (FS-LDM)[3] or the IBM Banking Data Warehouse (BDW) model[4]. These data models are built in a way that they combine regular modeling techniques with a meta-modeling approach for areas, where business requirements change frequently.

For example, business KPIs (e.g., profitability figures or other calculated data) and product features (e.g., interest rates and interest calculation methods) are not modeled as individual attributes but stored as code/value pairs, i.e. using a single value attribute combined with a code attribute that identifies the actual business attribute stored. The advantage of this approach is that new business attributes can be added (e.g., when a new product is introduced) without changing the logical or physical data model – existing load jobs will run unchanged. More thorough examples are given in the next subsection.

2.2 Practical Examples from the Financial Services Industry

As mentioned before, the authors have worked on various projects in different banking groups in Central and Eastern Europe, some of them based on the Teradata FS-LDM, of which a small simplified excerpt is shown in figure 2. Basically, the main entities shown are *PARTY* (representing customers, employees, but also organizational units) and *AGREEMENT* (representing all sorts of accounts and other contracts a customer may have with the bank).

As mentioned before, those areas of the data model dealing with less dynamic business requirements are modeled using a "traditional" approach, representing each business attribute as an attribute in the logical data model. An example is the customer master data (for individuals in the *INDIVIDUAL* and *INDIVIDUAL NAME HIST* entities, for orga-

[2]It turns out that these requirements are to some degree conflicting. This will be discussed in detail in the next subsection.

[3]http://www.teradata.com/t/logical-data-models/financial-services/

[4]http://www.ibm.com/software/data/industry-models/banking-data/

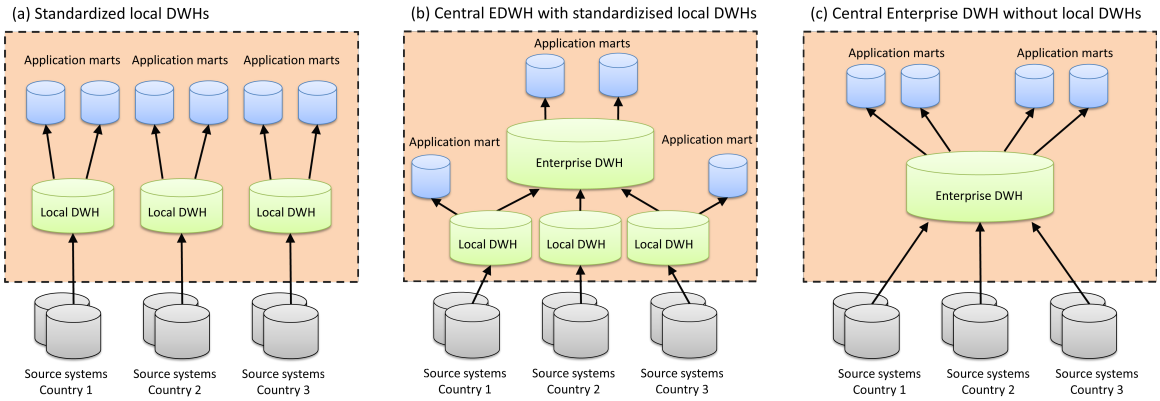

Figure 1: Data warehouse architectures for distributed organizations

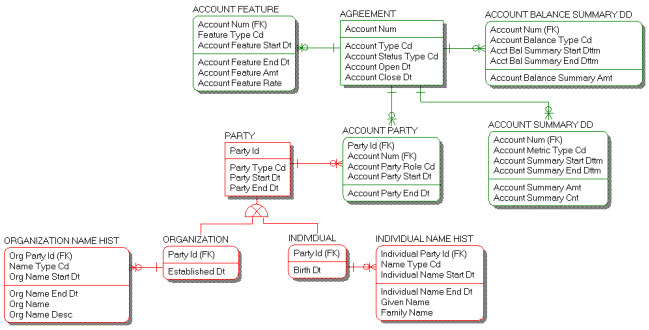

Figure 2: Excerpt of the Teradata FS-LDM

nizations in the *ORGANIZATION* and *ORGANIZATION NAME HIST* entities).

More dynamic business requirements are represented using a meta-modeling approach. For example features of an account (e.g. interest rates, nominal amounts) as well as derived KPIs (e.g. balances and other summary data) can be found as code/value pairs in the *ACCOUNT FEATURE*, *ACCOUNT BALANCE SUMMARY DD* and *ACCOUNT SUMMARY DD* entities. To cater for the different data types that may be used for different features, etc., a value attribute for each data type (e.g. *Account Feature Amt* for amounts, *Account Feature Rate* for rates) is used.

Now, recall the requirement to use the DWH data model to enable the standardization among different entities with different source systems. Basing the standardization solely on the LDM is not sufficient. As the business semantics are not fully covered by the model, different teams would map source data differently to the LDM.

The following are some real-world examples of problems, that can occur:

- **Same information (from different source systems) mapped differently to the DWH LDM/PDM.** For example, the current outstanding amount of a loan may be mapped to *ACCT SUMMARY DD.Account Summary Amt* by one team and to *ACCOUNT BALANCE SUMMARY DD.Account Balance Summary Amt* by another team. Actually, it can be discussed, whether the outstanding amout is a balance or not. As a result, an integrated use of the data (resp. reuse of mappings and transformations) becomes impossible.

- **Same information mapped to the same tables, but different codes used.** For example, the nominal interest rate of a loan may be correctly mapped to *ACCOUNT FEATURE.Account Feature Rate*, but one team may use 'ITR_NOM' as *Feature Type Cd*, while the other mapping team may use 'ITR'. Again, as a result the possibility of reuse becomes limited.

- **Different information mapped to the same place in the DWH.** While the first two points appear if semantically identical information is not recognized as such and thus not mapped consistently, the opposite can also occur. For example, there may be different days-past-due (DPD) counters, counting how many days a loan debt has not been paid according to the installment plan. There are different (all valid) business definitions, e.g. if holidays should be counted. If information from different sources with different business definition is mapped to the same place in the DWH (e.g. *ACCOUNT SUMMARY DD.Account Summary Cnt*using 'DPD' as *Account Metric Type Cd*, using this information in different application data marts will lead to unexpected results.

- **Different domain values used.** For example, the status of an account may be unambiguously mapped to *AGREEMENT.Account Status Type Cd*, but one source may use the value 'O' for open and 'C' for closed, while another may use 'A' for active and 'I' for inactive accounts. If this difference is not identified (and dealt with in the source to DWH transformation), the reusability of the data is again limited.

- **Lineage is only captured on LDM/PDM attribute level (not on code level).** Say, the nominal interest rate is stored in a data mart field *NOM_INTEREST_RATE*. It is mapped to *ACCOUNT FEATURE. Account Feature Rate*, which in turn is mapped to various source fields (i.e. various features with different

codes). In this case, the traceability of data requirements resp. application layer attributes to source attributes is ambiguous and thus limited.

Basically, in order to provide a solid basis for standardization, the LDM needs to be combined with a catalog of the codes used in those areas of the data model that are following a meta-modeling approach. Furthermore, it turns out that DWH LDMs, which also include a number of technical attributes to deal with data history, etc., are hard to comprehend by business users. As a consequence, they are of limited suitability to address the traceability requirement mentioned above. Business users will not be able to easily locate their data requirements in a catalog of LDM entities and attributes.

This all calls for a semantic business information model on top of the LDM to be used as an anchor point for the mappings and for communication with business users. Section 4 will discuss our approach how to build such a model as well as the use of semantic technologies to provide the necessary traceability.

3. RELATED WORK

As a matter of fact, we cannot state that traceability and data lineage are new issues; in the literature we can find various research approaches and published papers dating back to the early 1990s with proposed methodologies for software traceability [12]. The problem of data lineage tracing in data warehousing environments has been formally founded by Cui and Widom [3]. Moreover, based on the AutoMed notion of data lineage tracing, Fan and Poulovassilis [5, 10] developed algorithms for deriving affected data items along the transformation pathway.

These approaches formalize how to trace tuples (resp. attribute values) through rather complex transformations, given that these transformations are known on schema level. In practice, this assumption often does not hold. Transformations may be documented in source-to-target matrices (before they are implemented, the so-called specification lineage) and implemented in ETL tools (the so-called implementation lineage). Our work concentrates on how to properly define this specification lineage, which is a huge problem in large-scale DWH projects, especially if different sources have to be consistently mapped to the same target. The contribution of this paper is the introduction of a business information model as the central mapping anchor point.

From a commercial product perspective, data lineage and impact analyses are usually provided by metadata repositories. There are two categories of such metadata management tools. Firstly, most ETL software products (e.g. Informatica PowerCenter[5], IBM DataStage[6], Microsoft SQL Server Integration Services[7]) include a repository storing the metadata created and managed by the tool. Most of those vendors also provide interfaces to import metadata from database and BI tools. A prominent example of this category is the IBM Metadata Workbench[8], part of the InfoSphere Information Server product family, which allows tracing the transformations implemented in DataStage. The second group of commercially available tools are specialized products which are not related to a particular ETL tool and thus need to import the metadata using standard Common Warehouse Metamodel (CWM)[9] or tool-specific interfaces. They are of course less tightly integrated and may therefore have limitations in the metadata they can import from a particular tool. On the other hand, they are usually more flexible in specifying a customized metadata model tailored to the specific requirements of an organization. Examples in this group of tools are ASG Rochade[10], Adaptive Metadata Manager[11] or Teradata Metadata Services (MDS)[12].

All these commercial tools meanwhile provide quite sophisticated mechanisms for data lineage and impact analyses based on the transformations extracted from the ETL tool (implementation lineage). However, they are rather limited in combining and comparing this information with business information models, mapping specifications (specification lineage) and data requirements from a requirements management process. A first promising (but in practice also limited) approach is the one provided by the IBM Metadata Workbench (see above) in combination with the other Information Server components Business Glossary[13] (providing access to a catalog of business terms and categories) and FastTrack[14] (a mapping tool which allows generating DataStage jobs from a mapping specification). Similarly, ASG metaGlossary[15] provides mechanisms for defining and categorizing business terms and mapping them to physical data elements, based on the Rochade repository (see above).

The use of (business) models as a basis for development is a well-known concept in the software engineering world, referred to as Model-Driven Engineering (MDE) [15]. The most prominent MDE approach is OMG's Model-Driven Architecture (MDA) initiative[16], which defines system functionality using a platform-independent model (PIM). In fact, our business information model as presented in section 4 is such a PIM. Similar to MDE approaches in software engineering, we aim at deriving the development specification (source-to-target matrices) from this model.

Various research projects have been presented based on a mediated DWH architecture with an ontology infrastructure, providing ontology-based specification of relationships

[5] http://www.informatica.com/products_services/powercenter/
[6] http://www.ibm.com/software/data/integration/datastage/
[7] http://www.microsoft.com/sqlserver/
[8] http://www.ibm.com/software/data/infosphere/metadata-workbench/
[9] http://www.omg.org/cwm/
[10] http://www.asg.com/products/product_details.asp?code=ROC
[11] http://www,adaptive.com/products/mm.html
[12] http://www.teradata.com/t/tools-and-utilities/meta-data-services/
[13] http://www.ibm.com/software/data/infosphere/business-glossary/
[14] http://www.ibm.com/software/data/infosphere/fasttrack/
[15] http://www.asg.com/products/product_details.asp?code=AMG
[16] http://www.omg.org/mda/

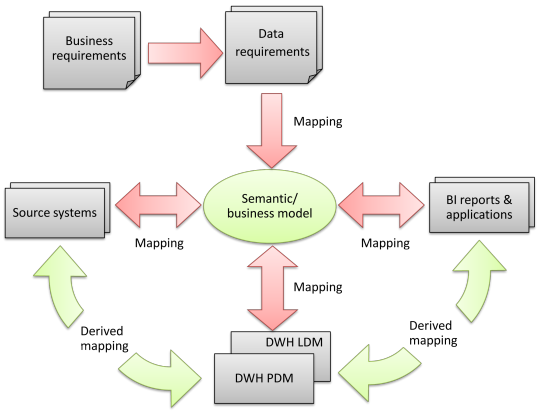

Figure 3: Business information model as the central mapping anchor point

between sources and the DWH on a conceptual level. For example, Romero and Abelló [14] address the design of a DWH multidimensional analysis schema starting from an OWL ontology that describes the data sources. They identify the dimensions that characterize a central concept under analysis (i.e. fact) by looking for concepts connected to it through one-to-many relationships. The relation between requirements, multidimensional design and underlying data sources were addressed by Mazón et al. [9] which applied a goal-oriented approach to requirements analysis for DWHs based on the TROPOS methodology. In [6] Giorgini et al. introduce an approach where conceptual multidimensional models capturing the various user requirements can be obtained. All three approaches are however not suitable for non-dimensional (ER-modeled, normalized) EDWH.

To the best of our knowledge, in current approaches, the use of semantic inference mechanisms for optimizing data lineage and impact analyses is not considered. Furthermore, there is still no comprehensive approach to support the representation and analysis of links over multiple layers of complex DWH architectures that would include conceptual business terms and data requirements. End-to-end tracing is however critical to enable successful reuse of mappings and transformations. Moreover, there are little resources on semantic mismatches and disparate use of terminology across DWH environments. In this context, ontologies can be used to represent the precise semantics that are required to support harmonization and ensure reusability.

4. SOLUTION OUTLINE

In section 2 we have shown that a DWH LDM or PDM alone is not suitable to cope with the traceability and standardization requirements to enable reuse in complex DWH environments. In order to address those requirements properly, the DWH data model has to be accompanied by a semantic or business information model. This model needs to capture the definitions and business rules plus the mappings to the different representations of the corresponding data artifacts.

The semantic model should provide a comprehensive and unique list of business concepts (with their attributes) needed to satisfy the information needs of all relevant user communities. As shown in figure 3, it is intended as an anchor point for all mappings. The semantic model should act as:

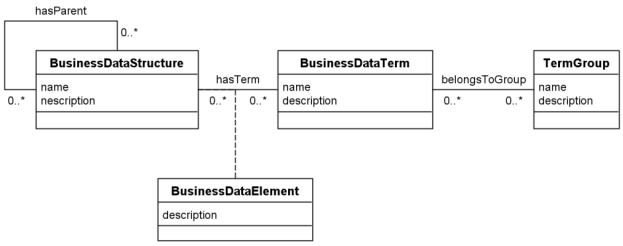

Figure 4: Business Data Elements (BDE) model

- **The main source for business-related metadata.** It should enhance the business users' understanding of the data by communicating the meaning and context, achieving traceability between the business concepts and their physical representations.
- **The main source to generate the DWH development specification.** As you can see in figure 3, by mapping the semantic model elements to the data sources, the DWH and targets such as application marts or semantic layers of a BI tool, the actual development specification for the transformations, the source-to-target matrices (STMs) can to a large degree be derived automatically.
- **The foundation for requirements management.** The model should show, which business concepts are already included in the DWH, which are requested but not yet implemented, who requested them, etc.
- **The foundation for enterprise information integration, enterprise data quality initiatives and increased usability.** It should facilitate consistent data modeling, consistent data definitions, allow to determine data redundancies, etc.
- **The foundation for improved time-to-market.** It should help to leverage synergies and enable higher development efficiency by reusing mappings, definitions, transformation jobs and analytical applications. It should reduce complexity by decomposition, etc.

4.1 Business Data Element Concept by Teradata

As an answer to the need for a semantic or business model, Teradata has recently defined a concept called Business Data Elements (BDE). The BDE concept is based on the metamodel shown in figure 4. Objects of interest to the organization are captured as a *BusinessDataStructure*, organized in a parent-child hierarchy – similar to the entity construct in the entity relationship model. Organization-wide unique business terms are captured as a *BusinessDataTerm*, in fact these are similar to the attribute construct in the entity relationship model. Finally, a *BusinessDataElement* is the intersection of a term in a given structure. In this regard, the main difference to the entity relationship model is the fact

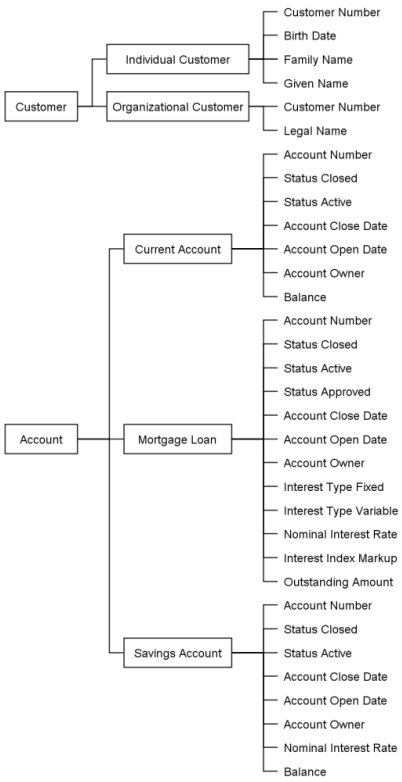

Figure 5: Simplified example BDE model for a bank

that the same term can be linked to more than one structure, i.e. uniquely defined terms can be reused in different structures independently from their hierarchy. Additional a *TermGroup* can be used to create user-specific views to the BDE model (e.g. a group of Basel II relevant BDEs).

Figure 5 shows a (simplified) excerpt of an example BDE model from the banking industry using the main structures *Customer* (with substructures *Individual Customer* and *Organizational Customer*) and *Account* (with substructures based on the products the bank offers, in this case *Current Account*, *Savings Account* and *Mortgage Loan*). Recalling the problems stated in section 2.2, note that *Outstanding Amount* and *Nominal Interest Rate* are explicitly listed as business terms. Also note, that *Status Active* and *Status Closed* are explicitly included (rather than a single *Status* term) in order to capture the different status values.

Now, how can such a BDE model be developed for a specific organization? A business model is always organization-specific, as the terminology and KPIs used will differ from organization to organization, especially when spanning country borders. Experience shows that even within the same language area (e.g. two German-speaking subsidiaries of the same banking group) terminology may differ significantly. Nevertheless, there are a number of sources than can help define the organization-specific BDE model.

In the case of Teradata, the main source of course is the reference LDM of the respective industry (providing explicit constructs for traditionally modeled business areas and example codes for areas that are represented using a meta modeling approach, see section 2), or rather the version of it that has been customized for the organization (mainly by omitting irrelevant parts). As shown in figure 6, further inputs that add content specific to the organization are corporate standards like a product data catalog and existing source data models – or legacy DWH or data mart data models in case of migration or data mart consolidation projects.

4.2 Ontologies and Semantic Technologies

As the connections between data elements in the different layers of a DWH system form graph structures, it suggests itself to utilize Semantic Web technologies which provide proper and mature methods and technologies for handling and operating on graphs. As we presented in [13], these technologies are highly suitable for performing lineage tracing operations in DWH environments based on (in [13] still solely implementation linage) metadata.

One of the basic concepts used in our approach and prototype (see next subsection), is the Resource Description Framework (RDF) [8], which uses a triple-based model (subject-predicate-object) meaning that resources have a number of properties with certain values. Values can be be again resources or literals. Resources and properties are uniquely identified by a Uniform Resource Identifier (URI). There is also a graphical notation, where an RDF document describes a graph having nodes (resources or literals) connected to each other via directed arcs (properties) and an XML-based syntax called RDF/XML which enables machines to process RDF metadata.

To enrich the representation formalism of RDF, the World Wide Web Concortium (W3C) introduced simple ontological modeling primitives (i.e. classes and subclasses) with RDF Schema (RDFS) [1]. In order to enable machines to fulfill reasoning tasks, stronger ontological conepts are needed. For example, one of our key concepts for tracing data elements is the transitivity of transformations through the different DWH layers. The Web Onlology Language (OWL) [4] extends RDFS by adding concepts for describing relations between classes (e.g. synonyms), cardinality and characteristics of properties (e.g. transitivity).

To be able to infer information from the ontology automatically, e.g. implicit transformation arcs arising from the transitivity property, inference rules have to be processed by an inference engine or reasoner. In addition to the predefined

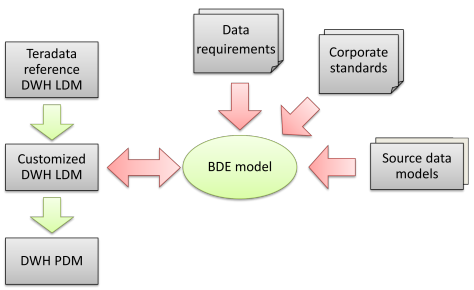

Figure 6: Creation of a specific BDE model

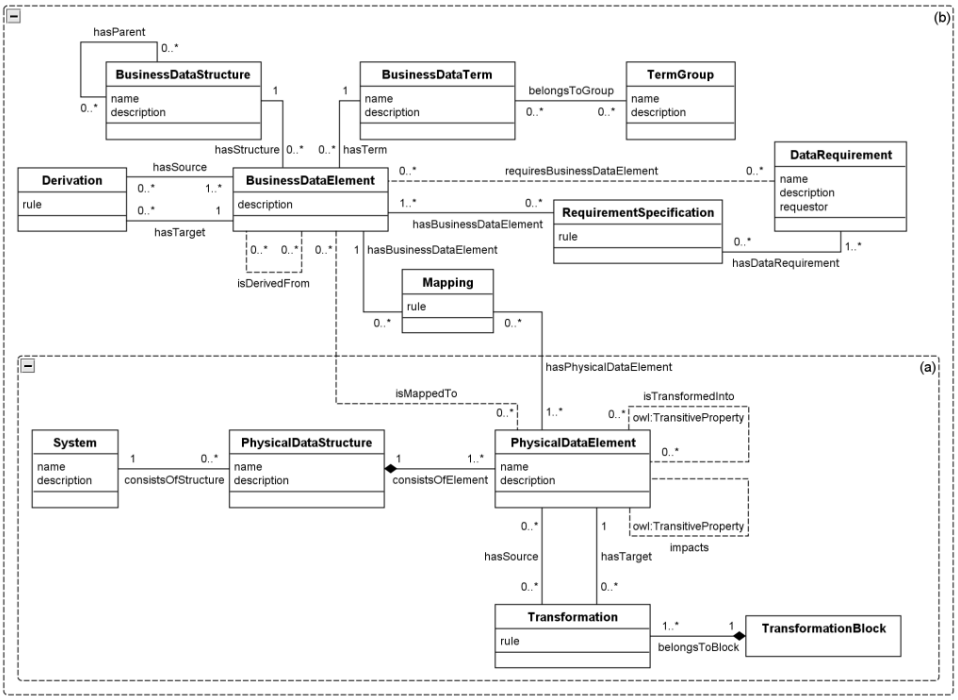

Figure 7: Prototype metamodel

inference rules in OWL, we use custom inference rules based on the Semantic Web Rule Language (SWRL) [7]. In order to efficiently extract information from the ontology we use the SPARQL Protocol and Query Language [11].

The suitability of Semantic Web Technologies in this environment may actually be questioned due to the open world assumption that they are based on. This means that formally, if a data element x does not have an *isTransformedInto* property to element y (see next subsection), we could not conclude that x is not transformed into y. But as we use the Semantic Web technologies in a closed environment with a closed repository, this is actually not an issue. We would even argue that an open world assumption is OK for most tracing problems (e.g. impact analysis, it is better to find an incomplete subset of impacted elements than none).

In addition to the suitability of semantic technologies to enable traceability among different data elements, ontology languages are a promising basis for extending the expressiveness of the business model. OWL offers a range of very useful constructs, like for describing synonyms or the disjunction of classes. Furthermore, there are domain ontologies freely available for different industries.[17] Representing the business model directly as an OWL ontology (i.e. business structures and terms as OWL classes and properties) is however subject to our future work (see section 5).

4.3 Research Prototype

In [13] we presented an approach and prototype utilizing Semantic Web Technologies to perform data lineage and impact analyses efficiently by infering additional traceability information from transitive properties and custom rules. Like most commercially available metadata management tools, this prototype was focussing on tracing physical data elements through transformations that may be implemented in an ETL tool (implementation lineage).

The part marked as (a) in the metamodel in figure 7 shows the coverage of the prototype that was already presented in [13]. As described in more detail below, it is implemented as an OWL ontology using the HP Jena framework[18] for persistent storage, inference and querying. In a nutshell, it shows physical data models and the transformations between elements within those models. A *System* consists of tables or flat files represented by the *PhysicalDataStructure* class which contains at least one *PhysicalDataElement* (columns in a table resp. fields in a flat file). The *System* construct can also be used to represent different layers (e.g. stage, core, etc.) in a DWH. The inferred transitive relationship *isTransformedInto* states that one column or field is transformed into another one. This relationship is automatically derived by a custom SWRL rule based on the fact that the physical data elements are specified as source and target of a *Transformation*. Moreover, *Transformations* can have transformation rules (e.g. a WHERE clause to limit the source tuples selected, or the assignement of a constant value to a target field) and can be grouped in *TransformationBlocks*. The also automatically derived relationship *impacts* reflects the fact, that a modification being made to a certain element may have an impact on another element. The *isTransformedInto* and *impacts* properties can be used to easily perform data lineage and impact analyses as simple SPARQL queries [13].

[17] e.g. `http://www.fadyart.com` for financial services

[18] `http://jena.sourceforge.net/`

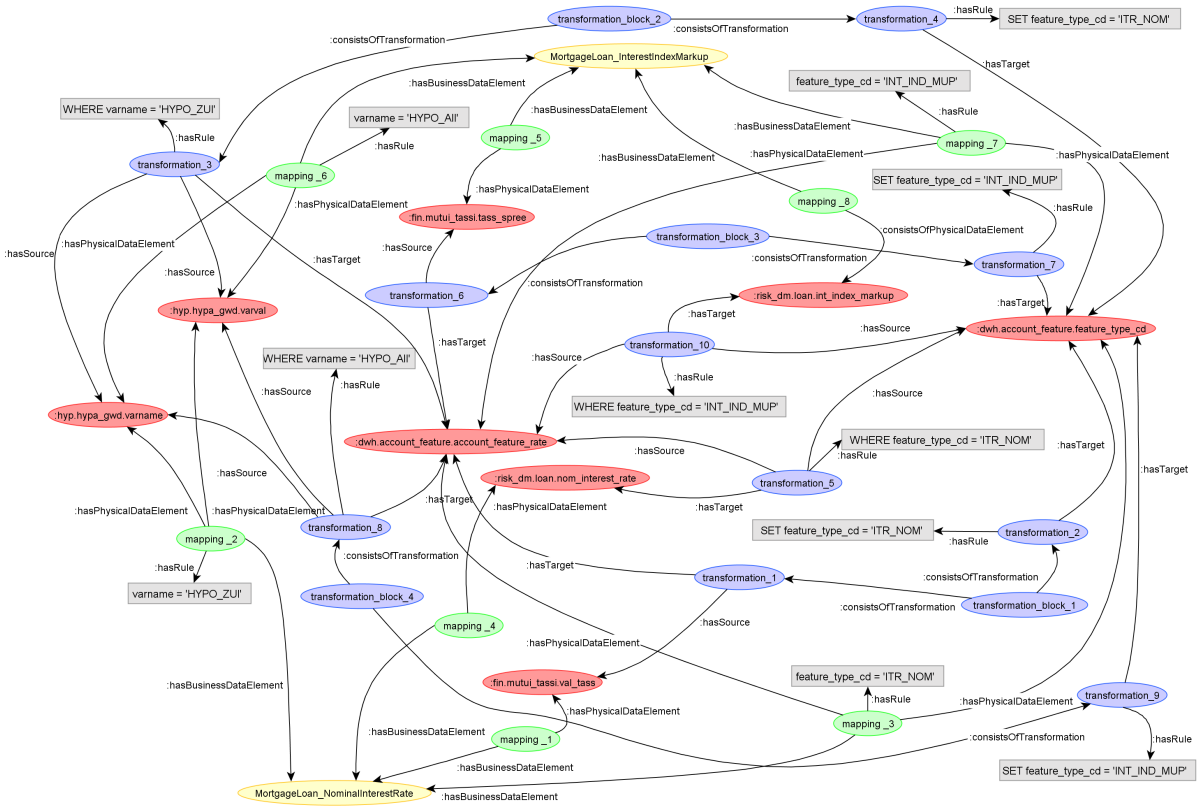

Figure 8: Example in RDF notation

Meanwhile, we have extended our prototype to also cover a business information model and data requirements as shown in figure 7 (b). This also distinguishes our approach from what most currently available commercial systems offer, which usually focus on the implementation rather than specification lineage (see section 3). Currently we have explicitly represented the business data elements metamodel (see section 4.1), i.e. business data structures become individuals of a class *BusinessDataStructure*, which can form hierarchical structures, and business data terms become individuals of a class *BusinessDataTerm*, which can be grouped in *TermGroups*. As future work, we will investigate using OWL classes and properties themselves to represent the business data structures and terms, i.e. represent the BDE model as an OWL ontology (rather than individuals of a metamodel represented in OWL).[19] The intersection of a business data term and a business data structure is represented as a *BusinessDataElement*. Note that the modeling is slightly different than presented in section 4.1 (*BusinessDataElement* is modeled as a regular class rather than an association class) in order to be able to represent this part in OWL.

Part (b) is linked to part (a) via the mapping of business data elements to a physical data elements. The relationship *isMappedTo* will be inferred if a BDE is connected to a physical data element via a *Mapping*, which can have mapping rules (similar to the transformation rules described above), along with the relationships *hasBusinessDataElement* and *hasPhysicaldataElement*. The fact that a BDE can be derived from other BDEs (according to a defined derivation rule) is reflected by the class *Derivation*.

Data requirements are represented using the class *DataRequirement* with a name, a description and the assignment of a requestor. Data requirements are formally specified by selecting one or more BDEs (using the *RequirementSpecification* class). Again, the relationship *requiresBusinessDataElement* will be derived for easier querying.

Figure 8 shows an excerpt of the resulting ontology of our banking example. It covers the *MortgageLoan_NominalInterestRate* and *MortgageLoan_InterestIndexMarkup* (i.e. the markup on a reference interest rate like Euribor) BDEs (shown yellow) and its physical representations (shown red) in two source systems (called *HYP* and *FIN*) and a data mart used for risk management (called *RISK_DM*). As discussed in section 2.2, *MortgageLoan_NominalInterestRate* is represented in the DWH as a *ACCOUNT_FEATURE.ACCOUNT_FEATURE_RATE* using 'ITR_NOM' as the *FEATURE_TYPE_CD*. Note that the inferred properties *isTransformedInto* and *impacts* are not shown in figure 8 for readability reasons.

Also note, that the view in figure 8 is not meant to be exposed to the users. Instead, figure 9 (a) shows the result set and the inferred *isTransformedInto* properties of tracing the physical column *RISK_DM.LOAN.NOM_INTEREST_-*

[19] Note that doing this properly is a non-trivial task as coupling the BDE ontology with the rest of our metamodel would require RDF reification mechanisms as different levels of modeling abstractions would be mixed.

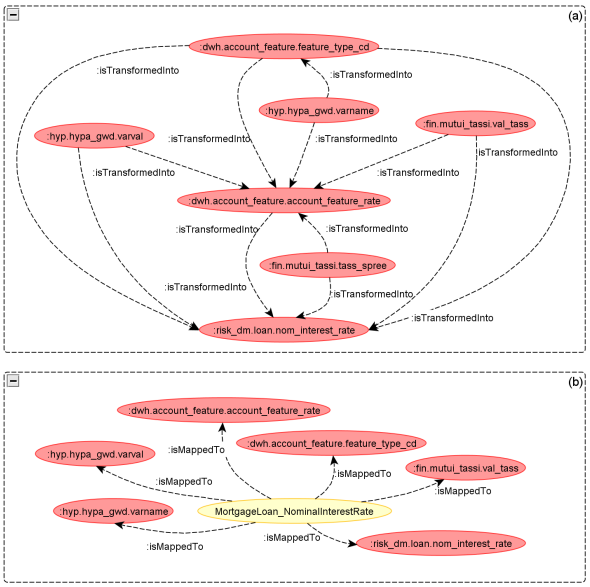

Figure 9: Tracability queries with derived properties

RATE back to its origin (based solely on the physical transformations, i.e. not considereing the business model). Note that the source column *FIN.MUTUI_TASSI.TASS_SPRE* is also shown, although it actually contains a different rate value (the *MortgageLoan_InterestIndexMarkup* BDE). It is detected by the lineage query as it is also stored in the same DWH column *ACCOUNT_FEATURE.ACCOUNT_FEATURE_RATE*, but using a different *FEATURE_TYPE_CD*. This is exactly the lineage ambiguity problem we have mentioned in section 2.2. In turn, figure 9 (b) shows the tracing information when a business model is used. In this case, the BDE *MortgageLoan_NominalInterestRate* and their physical representations are shown, using the mapping property *isMappedTo* rather than the physical transformations.

Finally, figure 10 presents the architecture of our research prototype. The representation of the presented metamodel in OWL is realized by using Protégé[20] and then imported into HP Jena, containing a rule-based inference engine capable of processing custom inference rules and mechanisms to store, manage and query ontologies [2]. The ontology is persistently stored in a MySQL DB. In order to define the mappings and transformations between data elements (both business and physical), we developed a simple mapping editor which creates RDF/XML code ready to be processed by Jena. Finally, we revert to the JUNG framework [21] for visualizing the query result sets as graph structures.

5. CONCLUSION & FUTURE WORK

We have shown that semantic business models are inevitable for enabling harmonization and reuse in complex DWH environments. The Teradata Business Data Element (BDE) concept can serve as a basis for the definition of such a model. Furthermore, we have presented an approach which utilizes Semantic Web standards and technologies to manage business as well as physical data elements and their connections through mappings and transformations. We have extended our prototype presented in [13] accordingly and introduced the notion of data requirements.

One further area of research is the definition of a proper foundation, rules and best practices for the semantic resp. BDE model. Besides investigating the use of OWL classes and properties to represent this model, main questions that remain answered are:

- **How to deal with the parent-child hierarchy of the business data structures?** Should business terms be assigned to all levels? What impact does this have on the transformations that can be derived from the mapping. For example, consider a the *Account Number* term from the example in section 4.1. Should there be only BDEs for the individual account types (i.e. *Current Account.Account Number*, *Savings Account.Account Number*, *Mortgage Loan.Account Number*) or only for the generalized account (i.e. *Account.Account Number*), or both?[22] There may be a source system that delivers data for all types of accounts, but also individual source systems with different mappings. Hence, this question is not trivial.

- **How should relationships between business structures be modeled?** Consider the business term *Account Owner* from the example. There could also be a corresponding *Owned Accounts* term linked to the *Customer* structure. In this case the two BDEs should be linked with a derivation rule. Another approach would be to explicitly model relationships as individual structures (i.e. introduce a *Account Customer Relationship* structure). Again, the impact on the usability for BDE-to-physical mappings and derivation of transformations has to be considered.

- **How should the mapping, derivation and transformation rules be specified?** In the example in section 4.3 we used SQL-like rules specifying the codes like 'ITR_NOM', etc. Further expressiveness will be needed for more complex rules. In order to allow the automatic derivation of physical transformations from the BDE mappings, this needs to be (at least semi-) formally defined.

- **How detailed should a business structure model be?** Or in other words, how many business structures are manageable and can be properly browsed by business users? So far we have worked in projects with business models containing 80-100 structures. It seems that this is a somehow practicable number, but this needs to be verified more thoroughly.

- **Which categorization schemes are practicable as additional term groupings?** The authors have experimented with both business categories as well as "technical" categories such as distinguishing master data (attributes) from measures, but further research is needed. Furthermore, it needs to be checked

[20] http://protege.stanford.edu
[21] `http://jung.sourceforge.net/`
[22] The use of OWL would actually automatically duplicate the properties down the hierarchy through inheritance.

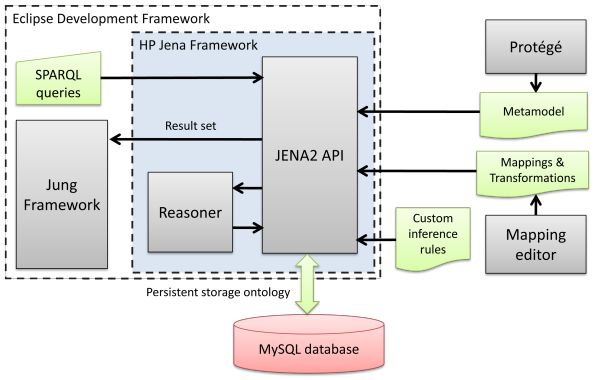

Figure 10: Prototype architecture

if the model needs to be enhanced to also include BDE groups (in addition to term groups).

- **How to setup governance for business data structures and terms?** Business structures should be aligned across the enterprise (similar to the entities in an an enterprise-wide ER model). Business terms (cf. the attributes in an ER model) can be defined iteratively as they get requested by a certain business department. Alignment then needs to take place as part of the iterative process to avoid redundancies. Corresponding data governance roles and responsibilities need to be defined in detail.

Another field of work is the definition of a proper tool support, both for BDE management and for defining source-to-BDE, BDE-to-DWH and BDE-to-target (e.g. data mart) mappings and to derive the development specification for the transformations. As mentioned in the related work in section 3, there are bits and pieces available on the market (e.g. IBM FastTrack with Business Glossary, Teradata Mapping Manager, etc.), but none of them fulfills all needed requirements. We will continue to extend our research prototype as a proof of concept.

Furthermore, based on the presented work, we plan to develop techniques to perform impact analyses on requirement changes in a timely manner to support DWH evolution and change management. Especially, the semantic representation and tracing of data requirements and their links to DWH architecture components (business and physical data elements, transformations) will be studied further. Moreover, we will examine if and how security requirements can be integrated into the semantic model and how traceability also for security requirements can be realized.

6. REFERENCES

[1] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema/, 2004.

[2] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proc. of 13th Intl. World Wide Web Conference on Alternate track papers & posters (WWW '04)*, pages 74–83, New York, NY, USA, 2004. ACM.

[3] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *The VLDB Journal The International Journal on Very Large Data Bases*, 12(1):41–58, 2003.

[4] M. Dean and G. Schreiber. OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004. http://www.w3.org/TR/2004/REC-owl-ref-20040210/,2004.

[5] H. Fan and A. Poulovassilis. Using AutoMed metadata in data warehousing environments. In *Proc. of the 6th ACM Intl. workshop on Data warehousing and OLAP (DOLAP'03)*, 2003.

[6] P. Giorgini, S. Rizzi, and M. Garzetti. Goal-oriented requirement analysis for data warehouse design. In *Proc of the 8th ACM Intl. Workshop on Data Warehousing and OLAP (DOLAP'05)*, pages 47–56. ACM, 2005.

[7] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004. http://www.w3.org/Submission/SWRL/, 2004.

[8] F. Manola and E. Miller. RDF Primer, W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer/, 2004.

[9] J.-N. Mazón, J. Trujillo, and J. Lechtenbörger. Reconciling requirement-driven data warehouses with data sources via multidimensional normal forms. *Data & Knowledge Engineering*, 63(3):725–751, 2007.

[10] A. Poulovassilis. Tracing Data Lineage Using Schema Transformation Pathways. In *Knowledge Transformation for the Semantic Web*, volume 95 of *Frontiers in Artificial Intelligence and Applications*, pages 64–79. IOS Press, 2003.

[11] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF, W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/, 2008.

[12] B. Ramesh and M. Jarke. Toward reference models of requirements traceability. *IEEE Trans. Software Eng* , 27(1):58–93, 2001.

[13] A. Reisser and T. Priebe. Utilizing Semantic Web Technologies for Efficient Data Lineage and Impact Analyses in Data Warehouse Environments. In *Proc. of the 8th Intl. Workshop on Web Semantics (WebS '09), in conjunction with DEXA '09*, pages 59–63, Washington, DC, USA, 2009. IEEE Computer Society.

[14] B. Romero and A. Abelló. Automating multidimensional design from ontologies. In *Proc. of the 10th ACM Intl. workshop on Data warehousing and OLAP (DOLAP '07)*, 2007.

[15] D. C. Schmidt. Guest Editor's Introduction: Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.