Applying a Knowledge Based System for Metadata Integration for Data Warehouses

Dan Wu and Anne Håkansson

Department of Communication System, School of ICT,
KTH Royal Institute of Technology,
Electrum 418, 164 40 Kista
{dwu,annehak}@kth.se
http://www.kth.se/ict?l=en_UK

Abstract. Data warehouses is a typical example of distributed systems where diverse tools and platforms need to communicate to understand each other. For the communication, metadata integration is significant. Seamless metadata interchange improves the data quality and the system effectiveness. Metadata standards exist, for instance, Common Warehouse MetaModel (CWM), which have enhanced the metadata integration. However, it is far from solving the problem of metadata integration in data warehouse environment. This paper proposes an approach to apply a knowledge-based system that supports the metadata integration. By utilizing the knowledge of software engineers on Common Warehouse MetaModels and the metadata interchange models, the knowledge-based system can give metadata interchange model suggestions. Such a knowledge-based system intends to partly automate the metadata integration to improve the efficiency and the quality of metadata integration in data warehouses.

Keywords: Metadata integration, Knowledge based system, CWM, Data Warehouse.

1 Introduction

A data warehouse (DW) is "a subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions[11]". In a DW environment, data is extracted from various heterogenous systems and integrated into the data warehouse for analysis purpose[8][11]. The Extraction, Transformation and Loading (ETL) tools automate the processes of data extraction, transformation and loading. DW is the target of the integrated data and applies multidimensional models, aggregation and selection applications on the data. The analysis tools such as OLAP and data mining apply further analysis on the data to generate results for the decision makers on different levels[8]. Since many tools, such as ETL tools, data modeling tools, DW repository tools and analysis tools, are involved in a DW environment, metadata integration is of critical importance to minimize the administration efforts and improve the extraction of information of DW[8].

R. Setchi et al. (Eds.): KES 2010, Part IV, LNAI 6279, pp. 60–69, 2010.

In the early 2000s, Object Management Group (OMG) adopted the Common Warehouse Metamodel (CWM) "to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments"[9]. CWM specification defines the metadata as "data about data. Examples of metadata include data element descriptions, data type descriptions, attribute/property descriptions, range/domain descriptions, and process/method descriptions[9]". CWM has defined an abstract language for expressing metadata models (called metamodels) for the various DW tools[9], and is supposed to be the solution for the metadata integration in the data warehouse systems. However, the DW repository vendors are slow in implementing CWM[10]. In many cases, systems have to be designed with diverse metadata for various reasons, such as "best-of-breed", "most compatible" or "most economic" [1]. In fact, it is hard to find robust tools that completely support CWM[1]. CWM does not serve its purpose as it is designed in practice and the metadata integration is often an ad hoc solution.

We assume that CWM is a good specification of metamodel for DW. The expert of CWM has a good understanding of general issues of metadata integration in DW environment. We therefore present an approach of a knowledge-based system (KBS) for supporting the metadata integration. The knowledge base in the KBS represents the knowledge of the expert engineers on the specification of CWM. The KBS also stores the instances of CWM metadata models and the translation of metadata models. The purpose of the system is partly automate the metadata integration process to improve the quality and the efficiency of the metadata integration by reusing the knowledge of metadata integration and accumulating the instances of the metadata models. The following sections are arranged as: Section 2 is about the related work around CWM and the metadata management in data warehouses; Section 3 is about the CWM specification and its implementing; Section 4 is the description of the knowledge-based system (KBS) model; Section 5 is the conclusions and the future work.

2 Related Work

The research on CWM is foremost concerned DW domain and for the DW system architecture modeling. Mazón et al.[6] describe a complete model driven approach for the development of all the components of DWs. The authors apply the UML and CWM to model a DW repository. Medina and Trujillo[3] show how to transform the multidimensional properties with the CWM package, Online Analytical Processing (OLAP), to a CWM-based metamodel, therefore to realize the metadata integration in a CWM-based DW system. Auth and Maur[4] describe a software architecture based on CWM with which to enable the metadata integration in the systems. CWM has inspired other areas of modeling, e.g., the enterprise modeling and the information mediation systems. Neaga and Harding[2] contribute with a common knowledge enterprise model using Model-Driven Architecture (MDA) and Common Warehouse MetaModel (CWM). Zhao and Siau[7] propose a new architecture established on the CWM resolving the

metadata problems for information mediation on the online information sources. While this paper suggests a new application with the CWM standard. The CWM specification is not only a model driven approach to DW systems, but also a source of knowledge on metadata integration for a KBS, which improves the metadata integration.

Another related area is the metadata management for data warehouses. Sen[1] who present a survey of the metadata management for the latest 50 years, proposes the concept of "metadata warehouse" to manage the metadata storing and the metadata changes. Zhao and Huang[12] describe how to use Description Logics to reason on CWM to improve the metadata consistencies and reduce metadata redundancies. Sen applied database technology in the implementation and Zhao and Huang tested their idea with the Description Logics. The research in this paper suggests the different approach of KBS and the focus of the KBS is the metadata integration.

3 Common Warehouse MetaModel Specification

CWM specification defines classes and associations abstracted from the common features and functions used in the DW domain. These classes and associations are organized in 21 packages and the packages are arranged in a 5-level-hierarchy. Each package represents a specific area in DW, e.g., relational resource. The architecture supports the model reuse and the understanding of the specification[9].

CWM conforms to Meta Object Facility (MOF), which is a standard technology for defining, constructing, managing, interchanging and integrating metadata[9]. MOF uses four levels architecture, named M3, M2, M1 and M0, where M0 is the lowest level with Object and M3 is the highest level with metadata object, see figure 1. CWM is an instance of MOF and a meta-metadata, so it is on the level M2. The models on M3 level describe the models on M2 level, the models on M2 level describe those on M1 level and the models on M1 level describe models on M0 level. The higher up the models are in the hierarchy, the more abstract the models become. Thus, the specific CWM metadata model instances are on the M1 level, describing specific models conformed to CWM. Our idea is to generalize CWM to integrate the models on M2 and M1 levels.

Meta-level	MOF terms	Examples
M3	Meta-MetaModel	The ”MOF” model
M2	MetaModel, Metametadata	UML metamodel, CWM metamodel
M1	Model, Metadata	UML models, CWM metadata
M0	Object, data	Modeled systems, Warehouse data

Fig. 1. OMG Metadata Architecture[9]

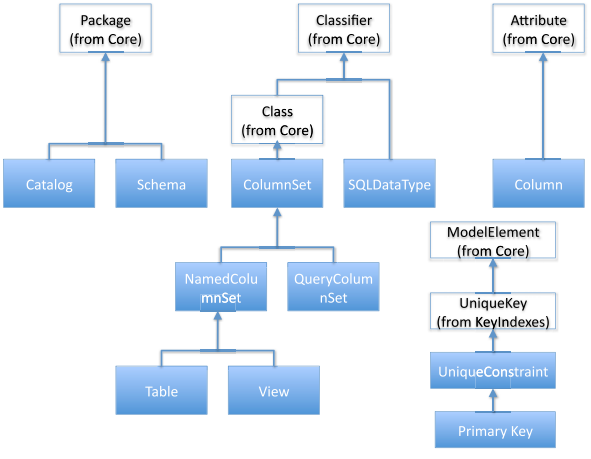

Fig. 2. Relational Package inheritances (partly) [9]

CWM uses the UML notation and the Object Constraint Language (OCL) to define the generic warehouse metamodels. By extending the Unified Modeling Language (UML), all the CWM meta-classes are inherited from the UML meta-classes. Figure 2 shows some of the relational package inheritances, e.g., the Schema inherits the Package and the Table inherits the Classifier.

CWM is designed for, e.g., DW tool vendors and DW end-users. figure 3 shows a CWM implementing instance that is developed by CWM users. Comparing with Figure 2, it shows how an instance follows the CWM specification. Figure 2 does not show the relationship between Schema and Table. However, in the Package of Core in CWM specification, it defines that a Package contains ModelElements such Packages and Classifiers. Therefore, the Schema Company contains several tables. Column inherits Attribute, which inherits Feature from the Core Package; Classifier is an abstract class and Feature is its contained elements. So the table Employees contains several columns. EmpKey is an instance of UniqueConstraint, an alternative is to define it as an instance of PrimaryKey. The CWM specification[9], the book[5] and the articles [3][4] are recommended for more detailed knowledge of CWM specification and its implementation.

4 The Approach of the Knowledge Based System

4.1 The KBS on CWM

The strength to integrate data warehouse metadata based on CWM metamodel is shown when data warehouse tool vendors implement the CWM specification. We define this as a passive solution. An active solution is that a KBS will apply the knowledge on CWM and the metadata integration to suggest models for the metadata integration. Our idea is that a KBS can present the definition and associations of model elements of CWM and store all the implementing

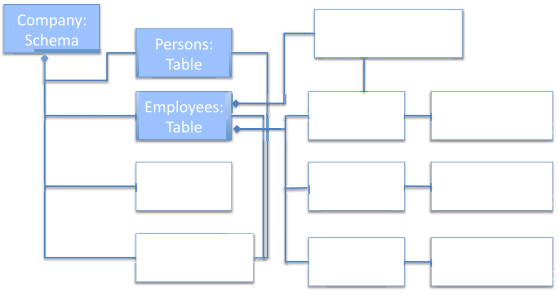

Fig. 3. CWM metadata instance of a relational database

instances. Thus the KBS can turn the CWM specification into a theoretical model, and turn the CWM implementing instances into heuristic cases. Thereby, the KBS will suggest metadata interchange models. The theoretical model of CWM will be used to verify the CWM implementing instance. The suggestions and verifications of the metadata interchange models are semi-automatic, hence change the passive status of CWM for the metadata integration.

The CWM specification is a generalization of the metadata of certain data warehouse tools. The sharing ability of information presented in CWM has been checked and refined by examining the metadata needs of several different, but representative, implementations as well as a broad range of representative warehouse configurations[9]. Therefore the KBS is designed to use logic rules to reason in two directions, i.e., from the specific metadata models to the generalization metamodel and vice versa. The reasoning is between the M2 and M1 levels in Figure 1. Moreover, with the help of the CWM implementing instances, the KBS can reason more efficiently. The CWM implementing instances can extend the definitions of the CWM metamodel because the CWM allows extending to most of the classes for specific model designs. By combining the CWM specification and the CWM implementing instances, a KBS is designed to support the metadata integration for CWM compatible and incompatible data warehouse metamodels. The KBS supports data warehouses to integrate metadata more easily and with more broader area of tools. Additionally, the KBS can improve the quality of the metadata management in general. The idea is illustrated in figure 4.

In figure 4, 'CWM based Metamodel Interchange Control' implements the CWM specification by describing the definition, the behavior and the associations of of classes in CWM specification. 'Metadata model A' is an instance of CWM. 'Metadata model translate between B and CWM' is an example of the translation of the metadata models made by the software engineers. With the knowledge of CWM specification and the collected instances such as A and B, the KBS is able to suggest the translations between CWM and the model C. The model of A, and the translated model of B and C are saved by the KBS and will enhance the KBS performance as the heuristic cases are increasing. Figure 4 also shows horizontal and vertical reasoning. The vertical reasoning includes

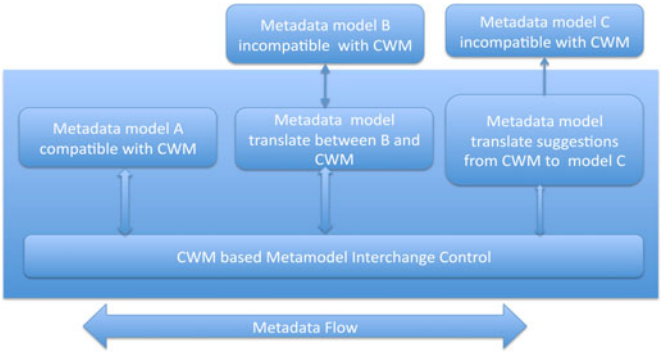

Fig. 4. The metadata integration idea of KBS

reasoning between the meta-metadata levels and the metadata levels (recalling M2 and M1 metadata levels in Figure 1), i.e., from CWM based metamodel to the specific models A, B and C and vice versa. The vertical reasoning checks, for instance, if implemented metadata models follow the CWM standard correctly. The horizontal reasoning is inside each level, for example checking the consistence between the CWM metamodels, and comparing the models among specific model A, B, and C.

The components of the KBS are shown in figure 5. The 'CWM metadata repository' stores the metadata models and the 'Metadata/metamodel model translated examples' stores the translated metadata models and the metamodels. The Knowledge base (KB) has the knowledge of the expert software engineers and can explain, reason and verify CWM metamodels and CWM instances. Ideally the KBS will be able to automatically suggest a CWM metadata interchange model.

The KB includes: definitions, associations and behaviors of CWM model elements; rules for generating CWM model instances; generalizing rules from a specific metadata model to metamodels; rules for checking consistence for levels of metamodels and the metadata models; rules for comparing metadata models and examining the similarity of the models; rules for extending CWM metamodels from the specific metadata models. With the KB, the system distinguishes elements by their associations, definitions and the behaviors. For example, Catalog in the relational package is the top level container. One of the Catalog's behaviors is to contain one or more Schemas. Schemas contain other elements, e.g. tables. The Catalog and the Schema interact as 'container' and 'ownedElement'. So KBS distinguishes Catalog and Schema by these definitions, elements' associations and behaviors. The 'relational package' supplies the context of this reasoning. Each package provides a reasoning context.

The goals of the KBS are: to implement a consistent CWM metadata management strategy; to distinguish CWM model elements and their instances by their definitions, associations and behaviors; to validate CWM metadata instances; to suggest the metadata interchange solutions to integrate data warehouse tools.

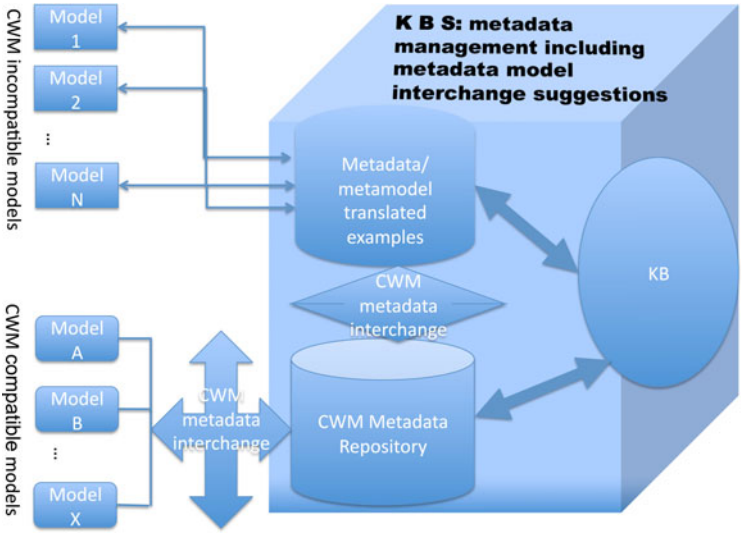

Fig. 5. The model of the Knowledge Based System

4.2 A Scenario

Whenever a data warehouse has changes in its data sources, targets or the business rules, a new metadata model might need to be integrated. To explain how the KBS can help with the metadata integration, a scenario is given. The UML diagram (in dark grey color) in figure 6 is a metadata model instance that needs to be integrated to CWM compatible models. The diagram in white color in the figure is an abstract description of the metadata model, i.e. metamodel of this model.

In the scenario, the KBS receives a task to translate a model of relational resource data into the CWM compatible system. The KBS compares it with the relational metadata instance as presented in figure 3 and the CWM metamodels. Then the KBS works to answer the following questions: (1) Is the Table and the Column the same as the CWM definition? (2) Is the PrimaryKey the same as the UniqueConstraint? (3) Is Datatype the same as the SQLSimpleType? (4) What is Domain and how to translate it into the CWM model?

To answer the first question, the KBS checks the definitions in the CWM specification and compare the definitions with the model and its metamodel. A simple rule is to assume that the same name of classes under the same context, i.e., the relational package, have the same definition, given that no conflicts between the classes are detected. To answer the second question, KBS follow the above mentioned simple rule, should suggest to translate the PrimaryKey to PrimaryKey in CWM. However, the question is to find the translation between the UniqueConstraint. KBS checks the definition of UniqueConstraint as "a condition to define uniqueness of rows in a table. An example of UniqueConstraint is a primary key[9]"; the definition of the PrimaryKey is "There is only

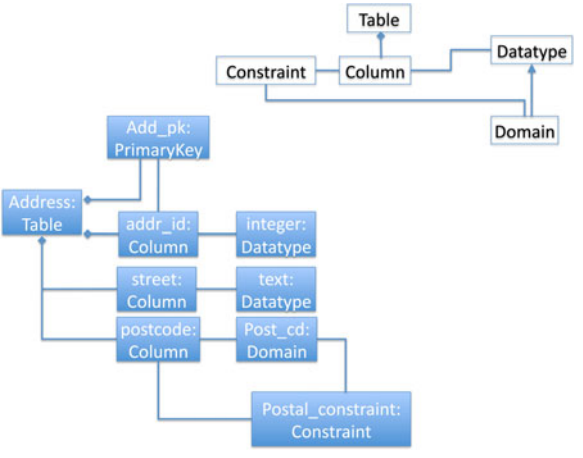

Fig. 6. An example of the CWM incompatible model

one UniqueConstraint of type PrimaryKey per Table." The CWM defines that UniqueConstraint is an ownedElement of a Table; and the UniqueConstraint is associated with the Column with UniqueKey Relationship. In figure 6, the PrimaryKey is an owned element of the table and associate to the column. Following these facts the KBS suggests to translate the PrimaryKey to an instance of the UniqueConstraint as it suggested in CWM. The KBS goes through the similar process to suggest that the Datatype can be translated into the SQLSimpleType. In the fourth question, the KBS meets the new metadata: Domain. However, the metamodel shows the relationship of the Domain to the Datatype, as well as to the Constraint, and to the Column, which are similar to the SQLDistinctType in CWM. The KBS searches the class of Column, since Column is the only known metadata related to the Domain. The KBS finds out the associations between the Column and the SQLSimpleType and the SQLDistinctType defined in CWM. KBS assumes that the Post_cd is either an instance of the SQLSimpleType or an instance of the SQLDistinctType. In this reasoning, the result from the third question, that the Datatype corresponds to the SQLSimpleType is used. Since Datatype is translated to the SQLSimpleType (result 3), the KBS assumes that the Domain relates more to the SQLDistinctType. Hence, comparing the relationship between Domain and Datatype and the relationship between SQLSimpleType and SQLDistinctType, the KBS suggests translating Domain to SQLDistinctType.

The KBS starts always from the closest concept of the models when reasoning. And then try to validate by detect logic conflicts with the integrated models. If no conflicts are found then the integrated model is verified and stored. The stored translated models are reused as heuristic instances in future reasoning.

5 Conclusions and Future Work

In this paper, we presented an approach of knowledge-based system that is applying the CWM specification and metadata model instances to support metadata integration between tools and systems. The KBS represents software engineers' knowledge of CWM and their experience of metadata interchange. The KBS also stores the translated metadata model instances. By reusing the knowledge and the stored metadata model instances, the KBS suggests metadata interchange models. Therefore, the KBS changes the ad-hoc solution into a semiautomatic process of metadata integration. The KBS also gives a better interaction with its users by explaining the suggestions of the metadata integration. With the support of the KBS, DW tool vendors and the data warehouse users will have more flexibility to choose best solutions without worrying about the metadata integration problem.

However, the KBS faces many challenges, e.g. the representation of the knowledge. The CWM specification generates many UML static models and XML files, which can be reused in the presentation of the knowledge. Hence, it is a difficult work to represent the Object Constraint Language (OCL), which is in text, with UML in the KBS. To reason from the abstract metamodel to a metadata model with the help of instances is also a challenge.

Another future work includes applying the KBS for CWM for metadata integration to other distributed systems, e.g., Internet. CWM is an example of metamodel standard. It would be very interesting to apply the KBS on other metadata standards and metadata model instances to solve the metadata integration problem.

Acknowledgements

This work is supported by Sweden-Korea Research Cooperation Program funded by the Swedish Foundation for International Cooperation in Research and Higher Education (STINT, http://www.stint.se/).

References

1. Sen, A.: Metadata management: past, present and future. Decision Support System 37, 151–173 (2004)
2. Neaga, E.I., Harding, J.A.: An enterprice modeling and integration framwork based on knowledge discovery and data mining. International Journal of Production Research 37(6), 1089–1108 (2005)
3. Medina, E., Trujillo, J.: A Standard for Representing Multidimensional Properties: The Common Warehouse Metamodel (CWM). In: Manolopoulos, Y., Návrat, P. (eds.) ADBIS 2002. LNCS, vol. 2435, pp. 232–247. Springer, Heidelberg (2002)
4. Auth, G., von Maur, E.: A software architecture for XML-based metadata interchange in data warehouse systems. In: Chaudhri, A.B., Unland, R., Djeraba, C., Lindner, W. (eds.) EDBT 2002. LNCS, vol. 2490, pp. 1–14. Springer, Heidelberg (2002)

5. Poole, J., Chang, D., Tolbert, D., Mellor, D.: Common Warehouse Metamodel, An Introduction to the Standard for Data Warehouse Integration. John Wiley & Sons, Inc., New York (2002)
6. Mazón, J.-N., Trujillo, J.: An MDA approach for the development of data warehouses. Decision Support Systems 45, 41–58 (2008)
7. Zhao, L., Siau, K.: Information Mediation Using Metamodels: An Approach Using XML and Common Warhouse Metamodel. Journal of Database Management 18(3), 69–82 (2007)
8. Staudt, M., Vaduva, A., Vetterli, T.: Metadata Management and Data Warehousing. University of Zurich (1999)
9. Object Management Group: Common Warehouse Metamodel (CWM) Specification, Version 1.1, vol. 1 (2003)
10. Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., Becker, B.: The Data Warehouse Lifecycle Toolkit, 2nd edn. Wiley, Indiana (2008)
11. Inmon, W.H.: Building the Data Warehouse, 3rd edn. Wiley, New York (2005)
12. Zhao, X., Huang, Z.: A Formal Framework for Reasoning on Metadata Based on CWM. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 371–384. Springer, Heidelberg (2006)