Advances in Intelligent Systems Research, volume 147
International Conference on Network, Communication, Computer Engineering (NCCE 2018)

ETL Transformation Function Modeling Approach Based on Model Transformation

Yu Zhou[1, a)] and Yue Zhang[2, b)]

[1] *School of Computer Science and Technology, Chongqing University of Posts & Telecommunications, Chongqing 400065, China;*
[2] *Software Engineering School, Chongqing University of Posts & Telecommunications, Chongqing 400065, China.*

[a)] 1920827591@qq.com
[b)] 1404152145@qq.com

Abstract. ETL is Extract-Transform-Load, which cleans, transforms, and integrates data according to uniform rules. At present, most of the methods for constructing ETL transformation function adopt a graphical representation method and lack precise semantic representation, which cause the computer to not automatically generate ETL process code. This paper proposes a modeling method of ETL transformation function based on model transformation. According to the model transformation idea in Model-Driven Architecture, Combining the semantics of relational metamodel which are accurately represented by the description logic SHIQ with the relational algebra representation method, to model the ETL transformation function from the relational schema data source to the multidimensional data warehouse. This method is of great significance for automatic code generation and improvement of ETL development efficiency.

Key words: ETL; SHIQ; Model-Driven; schema data; accurately represented.

INTRODUCTION

With the continuous development of data mining technology, Data Warehouse (DW) can effectively collect and transfer data scattered in different places into a unified data storage environment, which allows decentralized and inconsistent operational data to be efficiently translated into the required information. The Extract-Transformation-Load (ETL) process is the core of the business integration process and Data Warehouse (DW). It integrates by uniform rules and can improve the consistency and standardization of data1. DW is a collection of comprehensive topic databases chosen to support decision making2. Building a data warehouse involves the process of combining various forms of data, and provides a unified view of the data, and extracts data from different data sources and eliminate unwanted data3. Typically, the DW system consists of data source layer, ETL layer, storage layer for storage integration and aggregated data4.

Although the ETL process plays an important role in building and maintaining a data warehouse system, a standard model is lacking to represent the ETL process, which causes the ETL process to be designed, developed, and deployed every time the ETL process is implemented. It spends a lot of energy on the ETL team. In the literature 5and 6, four methods for constructing the ETL process are highlighted, they are Graphics-based ETL construction method, UML-based ETL construction method, Ontology-based ETL construction method and BPMN-based ETL construction method. After summarizing the above methods for constructing ETL processes, it was found that most of the ETL transformation functions are represented graphically. There are many limitations in this approach, which is inconvenient for ETL designers and developers to communicate, and it is not easy to implement code generation for the ETL process.

In order to solve the above problem, this paper proposes a model transformation method based on the ETL transformation function. ETL based on model transformation is to transfer the structured data sources and

unstructured data sources into data warehouses. This paper mainly studies the ETL transformation process from the data source described by the relational data model to the data warehouse described by the multidimensional data model. In order to accurately represent the semantics of the model, this paper represents relational data metamodel in a formal way. After getting the semantics of the model, the transform function rules and data mapping rules for defining ETL processes in conjunction with relational algebraic expressions. This approach helps ETL designers better design and develop ETL processes and makes ETL design process easy to build and maintain.

BACKGROUND KNOWLEDGE

MDA

The MDA (Model Driven Architecture) is an innovation of MDE (Model Driven Engineering). It uses UML as a modeling language and using OCL (Object Constraint Language) and Query/View/Transform as a model transformation language7.

The core idea of MDA is model transformation. The model transformation needs to define a series of transformation rules. The rules guide the source model transfer to the target model. The source model and the target model may not belong to the same meta-modeling level, and different model description languages may be used8.

MDA specifies the method of representing model that use metamodel to describe the model. The metamodel does not have explicit grammar and symbols. It is an abstract language that describes different types of data. In the MOF four-layer structure9, the metamodel corresponds to the M2 meta-model layer, and the meta-model contains the meta-elements used to construct the model. In MOF, the transition between models that occurs on the M1 model layer is essentially the mapping activity that occurs in the M2 model layer, that is, the mapping activity between the elements describing the source model and the elements describing the target model. Among them, the source meta-model describing the source model is mapped to the target meta-model describing the target model, the source meta-element is described to describe the target meta element, and the source meta-attribute is converted to describe the target meta-attribute.

THE CONCEPT OF ETL

It is known from the literature6, ETL scenario consists of extract data, transformation data, and load data. ETL data extraction is responsible for extracting data from different data source systems. Each data source has its own unique set of characteristics that need to be managed in order to effectively extract data from the ETL process.

ETL transformation data often performs a series of clean and validation of the input data, the purpose is to obtain correct and complete data. This process includes data cleansing, transformation and integration. It defines the granularity of the fact table, dimension tables, DW patterns, derived facts, slowly changing dimensions, and fact tables without facts.

ETL loading loads the transformation data into the target multidimensional structure, it includes loading dimension table and loading fact table. Dimension is specific angle of observation data and it is a type of attribute when considering a problem. The set of attributes constitute a dimension. The fact is the central theme of the multidimensional data model and it is composed of metrics. Metric is indicator of interest to decision makers and it is generally a quantization value of numerical value type.

DESCRIPTION LOGIC

Formalization methods include a variety of languages and technologies. Description Logic is a formalization language used to represent knowledge. It is often used to represent conceptual and conceptual hierarchie [11] and has deterministic reasoning capabilities10. In order to balance the expressiveness and complexity, this paper uses the SHIQ language in the SH family [12]. Besides the basic description logic function, it can also express quantitative restrictions, inverses, inclusions, and transitive relationships between relationships.

The Description Logic consists of Concepts, Roles, and Individuals. The set of individuals that satisfy a particular attribute is called a concept, and the binary relationship between concepts is called a role. There is an interpretation function $I := \left(\Delta^I, .^I\right)$ 11 in the Description Logic, where Δ^I denotes a non-empty interpretation

domain, and $\cdot^I$ denotes a functional interpretation domain. The semantics of the constructs and constructors in the Description Logic are represented by the interpretation function I mapping to set theory.

The knowledge base of Description Logic is composed of multiple Description Logic assertions. These assertions can be divided into two categories, namely, the Terminological Box (T Box) and the Assertion Box (A Box). Among them, TBox defines intrinsic knowledge, its basic form is to define the concept.

BUILDING ETL TRANSFORMATION FUNCTIONS

During the ETL transformation process, each ETL task can be configured through a custom function to achieve the design goals. Transformation functions are used to solve data coordination problems to ensure that the system can handle common and common data conflicts. Transformation functions generally fall into four categories: entity transformation functions, attribute transformation functions, user-defined functions, and structured transformation functions.

Entity Transformation functions can be used for data source tables, and attribute transformation functions can be used for data source attribute operations. A user-defined function is a function that can be added to the creator of an ETL scene. A structured transformation function is a function that can be used to convert unstructured (semi-structured and unstructured) data sources. It can convert semi-structured and unstructured data sources into structured data sources. The data sources discussed in this article are structured data sources and do not require discussion of structured transformation functions. The user-defined function can be extended to the entity transformation function according to a specific ETL scenario and can also be customized for the attribute transformation function.

Formalization of Relational Metamodels

The relational metamodel is the largest metamodel in the Common Warehouse Metamodel12 standard document. The CWM uses the package structure. Therefore, the elements in the relational metamodel must be able to reference to the object package and the core package. The relational metamodel describes metadata in the relational database, and also describes the interrelationships between meta elements. The relational metamodel has 24 meta element, which also depend on 44 meta element in other packages. In order to facilitate its presentation, this paper mainly describes the relational tables, columns, data types, and other elements of the relational metamodel in the CWM standard document, as shown in FIGURE 1.

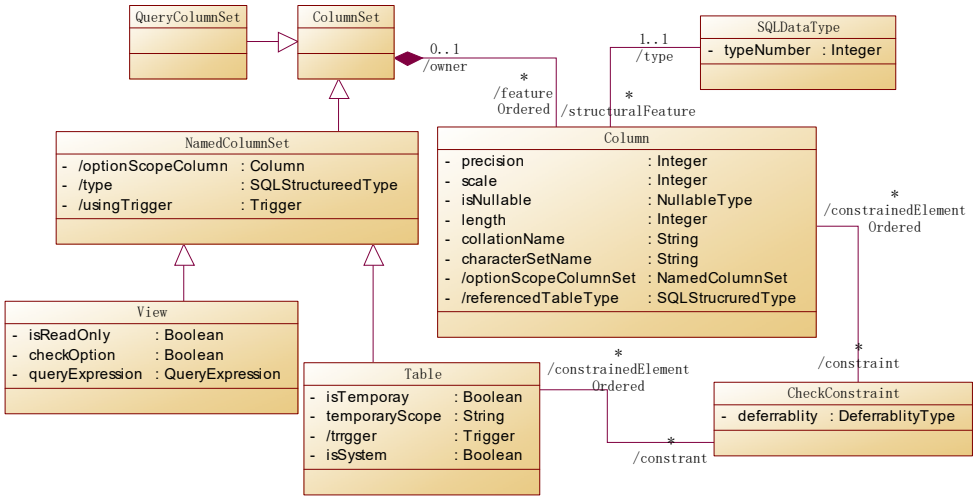

FIGURE 1. Table, Column and SQLData Type

There is a combination association between the meta element ColumnSet and the meta element Column, which means that the object instance of a ColumnSet is a combination of one or more Column object instances. The ColumnSet is the owner of the Column and the collection of the column is at least Composed of one column. The above description logic is expressed as:

$$ColumnSet \sqsubseteq 0\,feature.Column \tag{1}$$

The first-order logical representation of its corresponding semantics is:

$$ColumnSet^I \subseteq \left\{ x \in \Delta^I \mid \# \left\{ y \in \Delta^I \mid (x,y) \in feature^I \wedge y \in Column^I \right\} = 1 \right\} \tag{2}$$

There is a binary association between the meta element Column and the meta element SQLData Type, indicating that an object instance of a Column corresponds to an instance of an SQLData Type object, indicating that the data value of one column is bound to a corresponding data type. The above description logic is expressed as:

$$Column \sqsubseteq 1type.SQLDataType \tag{3}$$

The first-order logical representation of its corresponding semantics is:

$$Column^I \subseteq \left\{ x \in \Delta^I \mid \# \left\{ y \in \Delta^I \mid (x,y) \in type^I \wedge y \in SQLDataType^I \right\} = 1 \right\} \tag{4}$$

There is a combination association between a meta element ColumnSet and a meta element Column, there is a binary association between the meta element Column and the meta element SQLData Type, and a combination association between the meta element ColumnSet and the meta element SQLData Type can be deduced, indicating that the data value of a column is specified a corresponding data type. The above description logic is expressed as:

$$ColumnSet \sqsubseteq \forall Column.SQLDataType \tag{5}$$

The first-order logical representation of its corresponding semantics is:

$$ColumnSet^I \subseteq \left\{ x \in \Delta^I \mid \forall y (x,y) \in Column^I \rightarrow y \in SQLDataType^I \right\} \tag{6}$$

At the same time, there can also be a combination association between the meta element Table and the meta element SQLData Type, indicating that a column contained in a table is assigned a corresponding data type. Its description logic is expressed as:

$$Table \sqsubseteq \forall Column.SQLDataType \tag{7}$$

The first-order logical representation of its corresponding semantics is:

$$Table^I \subseteq \left\{ x \in \Delta^I \mid \forall y (x,y) \in Column^I \rightarrow y \in SQLDataType^I \right\} \tag{8}$$

Entity Transformation Functions

In the process of transformation between relational schemas, the mapping between the source schema and the target schema is accomplished through the entity mapping between the target schema and the entity schema in the source schema. Implementing a schema layer to generate a data map is a function that completes entity matching by defining a set of operations.

According to the formal representation of the relational metamodel in the previous section, the relationships between entities, attributes, and entities and attributes in the model layer M1 are explained.

Entity: Relationships R_1..., Rn and *I* are represented as tables in the relational data schema and are instances of the meta element Table in the relational metamodel. They are described as $Table(R_n)$ and $Table(I)$ using description logic.

Attribute: attribute C_1..., *Ci*..., *Cj*..., *Cn* represent columns in the relational data pattern and are instances of the meta element Column in the relational metamodel, and are described by the description logic as $Column(C_n)$

Relationship between entities and attributes: In the relational metamodel, the meta element Table obtains the combination relationship with the meta element Column through layer-level inheritance relations. According to description logic formula 7, the attribute set $\{C_1, C_2, ..., C_n\}$ contained in relation *R* is represented as $R(C_1, C_2, ..., C_n)$. $\{C_i, C_2, ..., C_j\}$ is a subset of $\{C_1, C_2, ..., C_n\}$.

Three types of algebraic operators are required pattern matching and data Transformation involved in the ETL transformation process: vertical operators, horizontal operators, operators that extend the instance for the target entity.

The first type of vertical operations includes Project, Join, and Rename operators.

Project. $I = \Pi_{C_i,...,C_j}(R)$ The projection operation is to use the entity *R*'s C_i..., C_j attributes and *R* to generate the same entity *I*, where $\{C_i, C_2, ..., C_j\}$ is the subset of properties owned by entity $R(C_1, C_2, ..., C_n)$.

Join. $I = R_1 \triangleright\triangleleft_P R_2$ The join operation is to generate a mode entity *I* that has all the non-connection attributes and connection attributes of the relationship R_1 and R_2. The *P* connection condition uses the attributes of the relationship R_1 and R_2. When *P* is true, R_1 and R_2 are Cartesian products.

Rename. $I = \rho_{C=C'}(R)$ The rename operation is to generate an entity *I* that contains all the attributes of the entity *R*, and the name of the attribute *C* in the entity *R* is changed to C'. Assume that $R(C, C_1, ..., C_n)$) changes the attribute name *C* in *R* to C'.

The second type of horizontal operations includes the Select, Union, Difference, and Deduplication operators

Select. $I = \sigma_P(R)$ The selection operation is to select the tuple or row satisfying the condition *P* from *R*. *P* represents a conditional expression that can be composed of variables, constants, and logical operators

Union. $I = R_1 \cup R_2$ The union operation is a union of all tuples of R_1 and R_2 to generate an entity, where R1 and R2 are two jointly compatible entities.

$$R_1 \cup R_2 = \{t \mid t \in R_1 \vee t \in R_2\} \tag{9}$$

Difference. $I = R_1 - R_2$ The difference operation requires that the attribute sets of R_1 and R_2 are the same, and the generated entity *I* contains only the instance of R_1 and does not include the instance type of R_2.

$$R_1 - R_2 = \{t \mid t \in R_1 \vee t \notin R_2\} \tag{10}$$

ED. $I = ED(R)$ The deduplication is eliminated the repeated tuples in the entity *R* to generate an entity *I*. Entity I cannot contain two identical tuples.

Attribute Transformation Functions

The matching of entities is accompanied by the matching of attributes between entities. In order to implement the mapping between entities, a set of operations is defined to complete the matching of attributes.

The mapping between attributes can also be classified using cardinality. The mapping between entities is represented by the mapping between attributes. To complete the mapping of the specified attribute, it is necessary to

complete the mapping between the attributes of the specified source entity and the attributes of the corresponding target entity. C_j represents the attribute of the target entity D, C_i represents the attribute of the source entity S, and the specific expression form is as follows:

$$MapAttribute\left(S.C_i, D.C_j\right) \tag{11}$$

The transformation between the source entity and the target entity must have the transformation between the attributes of the source entity and the attributes of the target entity. The transformation between attributes is included in the transformation of entity, which is a sufficient condition for entity transformation.

The transformation function of attributes can be classified according to unary operators and multivariate operators: unary operators are the operators that contain only one attribute value (for example, to String (), to Numeric (), etc.); multivariate operators contain at least two attribute values (for example, +, *, /, etc.). In multivariate operators, if the involved attributes are from the same entity, they are called ordinary multivariate operations; if the involved attributes are from multiple different multiple entities, they are called multivariate operations of set.

The unary operation performs transformation task. When two attributes are mapped to each other, the unary operation is required, but the two attributes come from different entities or domains. The expression for defining the transformation function of unary operation is as follows:

$$Operator1\left(S.C\right) \tag{12}$$

The transformation of data type is the transformation of one data type to another data type, which is based on different transformations of data type functions, such as "toString". The "toString(S.C)" changes the data type of the column in the data source, transforming the column's data type to a string data type, such as "toNumeric". The "toNumeric($S.C$)" changes the data type of the column in the data source, transforming the column's data type to a numeric type.

Ordinary multivariate operation performs transformation task. It contains a mapping between multiple attributes. The expression for defining the transformation function of ordinary multivariate operation is as follows:

$$OperatorN\left(S_1.C, S_1.C, ..., S_n.C\right) \tag{13}$$

The split of columns and the merging of columns are a pair of opposite operations, which are the most basic operations in the transformation operations of attributes.

Column Split. $Split\left(S.C, D.C_1, D.C_2, ..., D.C_n\right)$ The operations of the split of columns is to split the contents of a single column into multiple output columns. C represents a column in the source entity, C1, C2, ..., Cn represent the mapped column in the target entity.

Column Merger. $Merger\left(S.C_1, S.C_2, ..., S.C_n, D.C\right)$ The operations of the merging of columns is to combine the contents of multiple columns of different entities or the same entity into one output column. C represents the columns in the target entity that are output after merging, and C1, C2, ..., Cn represent columns that need to be merged in the source entity.

SUMMARY

First of all, this paper introduces the concept of ETL and the existing methods of constructing ETL process to represent the ETL transformation function. The relational data meta-model provided in the CWM public warehouse meta-model was studied, and the description logic language SHIQ was used to precisely represent the semantics of the tables, columns, and data types provided by the relational data meta-model. This paper introduces the idea of model and model transformation in MDA and uses MDA to represent the model. It uses relational data metamodel to describe the relational data source in the ETL transformation process. Using the idea of model transformation in MDA, the semantics of the relational data metamodel are described using description logic to define the rules of the transfer function of the ETL process, and the ETL transformation function is divided into entity transfer function

and attribute transfer function. The two transformation functions are described. In the future work, the ETL transformation function can be used to directly generate SQL code and automatically generate the ETL process code.

REFERENCES

1. Kimball R, Caserta J. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data[J]. Revista Brasileira De Oftalmologia, 2004, 71(3):199-204.
2. Kimball R, Ross M. The data warehouse toolkit: the complete gide to dimensional modeling[J]. Wiley, 2002, 6(3):181-191.
3. Kim H, Oussena S, Zhang Y, et al. Automatic Generation of Data Merging Program Codes[C]// Icsoft 2010 - Proceedings of the Fifth International Conference on Software and Data Technologies, Volume 2, Athens, Greece, July. DBLP, 2010:179-186.
4. Wojciechowski A. ETL workflow reparation by means of case-based reasoning[J]. Information Systems Frontiers, 2017:1-23.
5. Ali S M F, Wrembel R. From conceptual design to performance optimization of ETL workflows: current state of research and open problems[J]. Vldb Journal, 2017, 26(6):777-801.
6. El-Sappagh S H A, Hendawi A M A, Bastawissy A H E. A proposed model for data warehouse ETL processes[J]. Journal of King Saud University ¨c Computer & Information Sciences, 2011, 23(2):91-104.
7. Sacevski I. Introduction to Model Driven Architecture (MDA)[J]. Seminar Paper, 2007
8. Lano K, Kolahdouzrahimi S. Model-transformation Design Patterns[J]. Software Engineering IEEE Transactions on, 2014, 40(12):1224-1259.
9. Object Management Group. formal/2016-11-01. OMG Meta Object Facility (MOF) core specification version 2.5.1[S]. http://www.omg.org/spec/MOF/ 2.5.1/ 2016-11-01
10. Shi Lian Sun Ji-Gui. Descrption Logic Survey [J], 2006,33(1): 194-197.Berardi D, Calvanese D, Giacomo G D. Reasoning on UML class diagrams[J]. Artificial Intelligence, 2005, 168(1–2):70-118.
11. Peng Li, Yang Heng-fu. ABox Consistency Decision Algorithm for Description Logic SHIQ [J], 2013,39(12): 308-315.Baader. The Description Logic Handbook[J]. Kybernetes, 2003, 32(9/10):43-95.
12. Object Management Group. formal/03-03-02. Common Warehouse Metamodel (CWM) specification, version 1.1[S]. http://www.omg.org/spec/CWM/1.1/ PDF/2003-03-02.