Transforming UML Class Diagrams into HBase Based on Meta-model

Yan Li, Ping Gu, Chao Zhang
Department of Computer and Electronics and information
Guangxi University
Nanning, China
e-mail: ly.7437@163.com

***Abstract*—In order to reduce the influence of requirement change for software development and improve the efficiency and portability of software development efficiently, this paper, based on the ideas of Model Driven Architecture (MDA), proposes a method that transforms UML class diagrams into HBase based on Meta-model. The method achieves the transformation from Platform Independent Model (PIM) to Platform Specific Model (PSM) on the meta-model level and is comprised of three phases. In the first phase, the meta-models of UML class diagram and HBase database are built. In the second phase, the mapping rules between the two meta-models are proposed. In the last phase, the UML class diagram is built and the HBase database model is generated by transformation. At last, the paper uses Atlas language to achieve a breakfast serving system to prove the feasibility of the MDA in the software development.**

Keywords—HBase; MDA; meta-model; NoSQL; model transformation

I. INTRODUCTION

Model Driven Architecture (MDA) proposed by the Object Management Group (OMG) is a method of software development [1]. MDA treats models as the center of the entire software development process and has defined four kinds of model: Computation Independent Model (CIM), Platform Independent Model (PIM), Platform Specific Model (PSM) and Implementation Specific Model (ISM). MDA not only separates functional description of the system and achievement of the system on specific platform by those models, which can reduce the degree of coupling between business and technology, but also improve portability of the system and interoperability between different platforms. The core technology of MDA is model transformation. So the models of MDA are throughout the entire software development process, and the code is generated by the model automatically in the end. If we use this development mode, we only need to alter the original model when the requirement changes. Then the system will rebuild codes which conform to the new requirement by automatic transformation of models. This can improve the efficiency of software development efficiently and provide reliable basis for later maintenance.

With the rise of Internet web2.0 site, traditional relational databases have begun expose many problems and disadvantages in data processing. While NoSQL can deal mass data and has flexible and high-quality data model. Those characters of NoSQL are fully satisfied with the requirements of web2.0. So it achieves more and more attention, and researches on transforming existed model into NoSQL database model is becoming a hot spot. Some researchers have proposed the method that transforms relational database model into HBase model which is a very popular NoSQL database nowadays, as in [2].

When designing a web system, if a NoSQL database model is built according to the requirement directly, the database may be different from the model required by the customers because that method ignores the business factors of systems and the affection of requirement for data [3]. So the paper uses the idea of MDA to search a method that transforms UML class diagrams into HBase based on meta-model. The method is comprised of three phases. In the first phase, the meta-models of UML class diagrams and HBase database are built. In the second phase, the rules of transformation from meta-model of UML class diagrams to meta-model of HBase database are proposed. In the last phase, the UML class diagram which belonged to PIM is built according to requirement analysis and the HBase database model which belonged to PSM is generated by transformation.

II. INTRODUCTION OF HBASE

HBase [4] is a sparse, distributed, multidimensional database based on column. As an open-source database of Hadoop [5], HBase is a very popular NoSQL database nowadays.

The tables of HBase store a series of row records. Every row record in HBase has a storable Row Key, a Timestamp and an arbitrary number of Columns. Row Key in accordance with the order of the dictionary is the unique identification of row records in a table. Timestamp is used to index each data cell which can contain multiple versions of the same data. Column is defined as that: <Family>: <Qualifier>. A Column Family must be created when creating a table and the data saved in the same Column Family are stored together on the file system. Qualifier represents a column of the Column Family. A Column

978-1-4799-3197-2/14/$31.00 ©2014 IEEE

Family contains an arbitrary number of columns which can be added into a family dynamically. So different rows of a table must have a Row Key and the same Column Families, but might have completely different columns. The data in HBase don't have data styles. They are stored according to binary system. The combination of Row Key, Timestamp and Family: Qualifier can ensure the value of a data cell uniquely. The data model of HBase is shown in Figure 1.

Row Key	Timestamp	Column Family	
		URI	***Parser***
r1	t3	url=www.taobao.com	title=
	t2	host=taobao.com	
	t1		
r2	t5	url=www.alibaba.com	content=
	t4	host=alibaba.com	

Figure 1. Data model of HBase

III. MODEL TRANSFORMATION BASED ON META-MODEL

A. PRICIPLE OF MODEL TRANSFORMATION

UML class diagram belongs to PIM; while HBase database model belongs to PSM. So achieving the transformation from UML class diagram to HBase database model is achieving the transformation from PIM to PSM. The methods that transform PIM to PSM are various, such as manual switching, model transformation based on rules, model transformation based on relational algebra etc. The method the paper researches is based on meta-model which is a widely using and easily to grasp method.

Meta Object Facility (MOF), a standard established by Object Management Group (OMG), is one of the core technologies of MDA [6]. MOF is used to describe a unified way of modeling constructs. The framework of MOF has defined four levels which are called M0, M1, M2 and M3 from bottom to up in turn according to the term of OMG. Meta-model that is model of model in M1 level is in M2 level. The method based on meta-model uses mappings between elements of different models in M2 level as transforming rules to execute transformation, which achieves the transformation of models in M1 level finally [7].

UML class diagram and HBase database model are in the M1 level. If we want to transform UML class diagram into HBase database model based on meta-model, firstly we should build the source meta-model which can describe the elements of UML class diagram accurately and the target meta-model which can describe the elements of HBase database model accurately. Then we should realize the mapping between source meta-model and target meta-model. Finally we use these mapping rules as transforming rules to realize the special transformation from UML class diagram to HBase database model.

B. BUILD THE SOURCE META-MODEL

The models of MDA can be described by special UML. So the paper uses UML class diagram to describe the source meta-model.

UML class diagram displays a set of classes, interfaces, cooperation and the relations between them [8]. The interfaces and cooperation don't have any mapping relations with databases generally, so the paper considers the class diagram mainly contains two elements: class and relation. The class mainly contains a set of attributes and operations. The Object Identifier (OID) is a special kind of attributes and the parameter is an important element of operation. The relations between classes contain association, generalization, realization and dependency, while aggregation and composition belong to special associations. Meta-model of UML class diagram is shown in Figure 2.

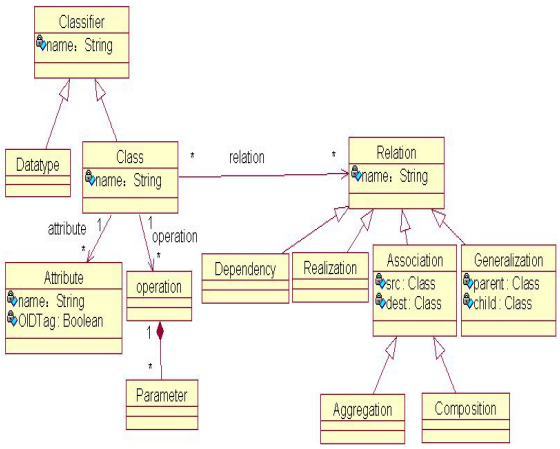

Figure 2. Source meta-model

C. BUILD THE TRAGET META-MODEL

HBase database includes five basic elements: table, row key, timestamp, column family and column. The meta-model of HBase database model is also described by UML class diagram in the paper, so the five elements of HBase database mode are mapped to five classes of UML class diagram. Table class represents database's tables which are formed by row key, timestamp and a set of column families. A column family has countless columns. Meta-model of HBase model is shown in Figure 3.

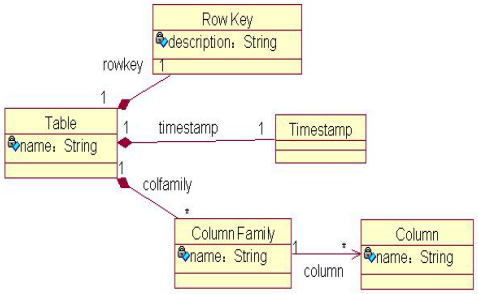

Figure 3. Target meta-model

D. RULES OF MODEL TRANSFORMATION

When the requirements of a web system have changed, the models will change. When requirement changes, in order to make sure we only need to alter the UML class diagram and the change can be reflected on the database model automatically, we should build the mapping relations between UML class diagram and HBase database model. So the paper defines the mapping relations between source meta-model and target meta-model built in section B and section C.

Class diagram describes the static structure of class in the system. It not only defines the class of system, but also contains the internal structure of class and relations between classes. According to the meta-model of UML class diagram, mapping rules can be divided into two parts: mapping rules of classes and mapping rules of relations.

1) MAPPING RULES OF CLASSES

Class which contains a set of attributes and operations is an important part of class diagram. The paper doesn't consider the mapping of operations, because operations have no direct mapping relation with the elements of database generally. So according to the meta-model of class diagram, mapping of this part mainly contains mapping of object identifier, mapping of attributes and mapping of classes.

a) Mapping OID to row key

In order to find the object we need in a system, every object would have its own identifying number which is called OID. OID is a special attribute of a class. The difference between OID and other attributes is OID is unique. However, row key is the unique identification of row records in a table of HBase. So the paper maps it to the row key of HBase. Row key can't be null, while OID of a class may not exist. So if a class doesn't have OID, adding a new attribute as OID or assigning a attribute existed as OID is necessary before that mapping.

b) Mapping attributes to columns

Attribute is used to describe the character of class. The all attributes except OID of class can be mapped to columns of HBase. Those columns are put into the same column family in the paper. As it's known that those columns which are often accessed together should be put into the same column family, but we don't know what attributes are related according to the class diagram. So column family should be redistricted after all the mapping.

c) Mapping classes to tables

Generally, classes are mapped to tables. That is building a table for every class in HBase database, and the name of every table is the name of corresponding class.

2) MAPPING RULES OF CLASSES' RELATIONS

According to meta-model of class diagram, classes have four kinds of relations: association, generalization, realization and dependency, while aggregation and composition are special association. Realization describes the relations between classes and interfaces, which has no mapping relation with the elements of database, while dependency has various representations, which is very complex. So the paper doesn't consider the relations of realization and dependency. The paper mainly researches the mapping rules of the relations of association, aggregation, composition and generalization.

a) Mapping of association relationship`

According to the difference of the number of objects involved by association, association relation can be divided into three kinds: "one to one" relationship, "one to many" relationship and "many to many" relationship. It's known that in order to reduce data redundancy, relational databases put the primary key of a table as foreign key into another associated table or add a new table used to store the primary keys of two associated tables according to the kind of association relationship. HBase is a high-capacity and column-oriented database whose column can be added dynamically and it doesn't support the join operation between tables. So the paper considers no matter what kinds of association relationship, two associated table should build a new column family which is used to store other's row key. The mapping is shown in Figure 4. As HBase don't support join operation and don't have bound of foreign keys like relational database, so HBase can't ensure the uniformity of data. The mapping the paper proposed are beneficial to ensure the eventually uniformity of data, although it adds data redundancy.

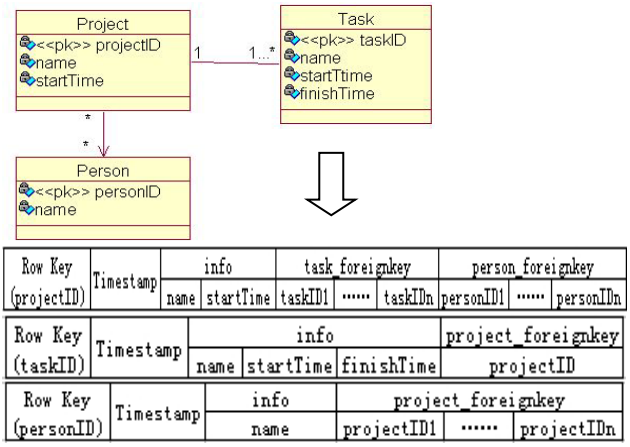

Figure 4. Mapping of association relationship

b) Mapping of aggregation relationship

Aggregation is a special association relationship. It represents two things exist whole/part relation which is weak. In the aggregation relationship, things belonged to "part" can be shared by many things belonged to "whole", and things belonged to "part" are independent of things belonged to "whole". So the paper considers aggregation as a ordinary association relationship, and the mapping rules of aggregation relationship are the same as the Mapping rules of association relationship. The mapping is shown in Figure 5.

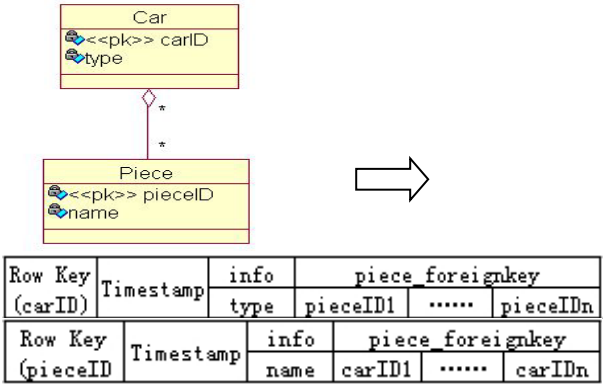

Figure 5. Mapping of aggregation relationship

c) Mapping of composition relationship

Composition is a special association relationship. It represents two things exist whole/part relation which is strong. In the composition relationship, things belonged to "part" can't be shared by many things belonged to "whole", and things belonged to "part" are depend on things belonged to "whole". So in the table which stores thing belonged to "part", the column which stores the row key of table which stores thing belonged to "whole" can't be null. In order to satisfy that condition, the row key of the table which stores thing belonged to "part" should be alter like that "partOID_wholeOID", and the table which stores thing belonged to "whole" should build a new column family which used to store the row key of the table which stores thing belonged to "part". The mapping is shown in Figure 6.

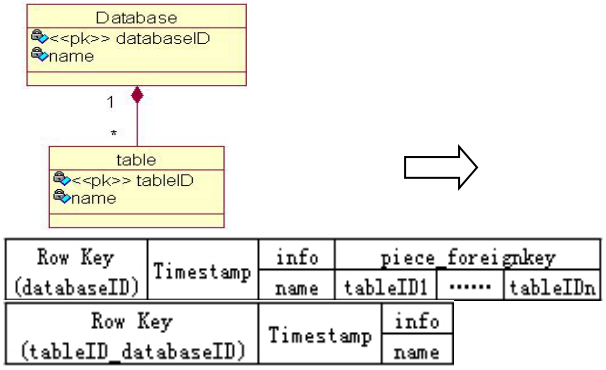

Figure 6. Mapping of composition relationship

d) Mapping of generalization relationship

Generalization is a kind of relationship between general description and detail description. It reflects the principle of classify and inheritance, that means a subclass can inherit all the attributes and operations of its parent class. So the paper maps all the attributes of subclass to the columns of table which corresponds to parent class, and deletes all the tables which correspond to subclass. What's more, the relations between subclass and other classes are transformed into the relations between root parent class and other classes. The mapping is shown in Figure 7.

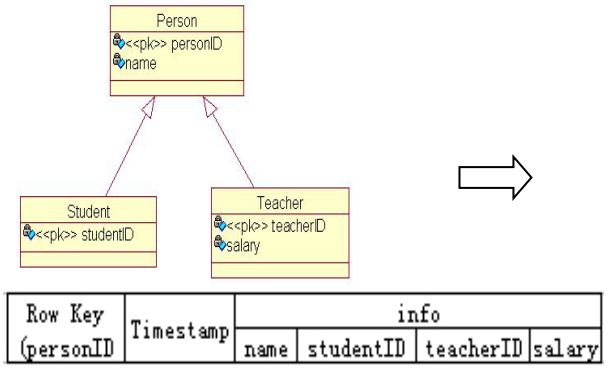

Figure 7. Mapping of generalization relationship

IV. IMPLEMENTATION OF MODEL TRANSFORMATION

Atlas Transformation Language (ATL) is created by the research group called ALTAS INRIA&LINA for answering to the OMG MOF/ QVT RFP [9]. It is a model transformation language specified as both a meta-model and a textual concrete syntax. ATL is a hybrid language which no only has the character of described language, but also contains the content of imperative language. It has provided a way of model transformation between source meta-model and target meta-model.

According to the hierarchical structure of ATL model transformation, an ATL program mainly contains five elements [10]: Ma which represents source model, Mb which represents target model, MMa which is a meta-model of Ma, MMb which is a meta-model of Mb and Mt which is an instance of model transformation. The transformation based on ATL only need to be provided source meta-model, target meta-model, rules about transformation between source meta-model and target meta-model, and the source model. Then it can generate the target model. The description of transforming rules this paper proposed using ATL is as follows:

Create tables: define rule Class2Table, and then transform all classes except subclasses into table. The key code is as follows:

```
rule Class2Table{
from
   a: UMLClass!Class(a.getgeneclass1->isEmpty())
to
   p: hbase1!Table (
        name<-a.name,
        rowkey<-r,
        timestamp<-t,
        colfamily<-CF,
        colfamily<-CompositionCF,
        colfamily<-AssociationCF ),
```

Create row key: transform the OID of the "class" or the combination of OID of the "class" and the OID of other class which has composition relationship with the "class" and represents thing belonged to "whole" into the row key of the table generated according to the "class". The key code is as follows:

```
r:hbase1!RowKey(
            name<-'('+a.getrowkey () +')'
   ),
```

Create column family and column: transform all the attributes of the "class", except OID, and all the attributes of subclasses whose parent class is the "class" into columns which are in the same column family of the table generated according to the "class". The key code is as follows:

```
CF: hbase1!ColumnFamily (
            name<-'info',
            column<-ColumnOfNoOId,
            column<-GeneralizationCol
   ),
```

Create column family and column: transform the OID of a class which has composition relationship with the "class" and represents thing belonged to "part" into columns which are in the same column family of the table generated according to the "class". The key code is as follows:

```
CompositionCF: distinct hbase1!ColumnFamily foreach
(class in a.getComclass2) (
       name<-class.name+'foreign',
       column<-CompositionCol
   ),
```

Create column family and column: transform the OID of a class which has association relationship or aggregation relationship with the "class" into columns which are in the same column family of the table generated according to the "class". The key code is as follows:

```
AssociationCF: distinct hbase1! ColumnFamily foreach
(class in a.getassoclass1) (
            name<-class.name+'foreign',
            column<-AssociationCol
)
```

The paper realizes the transformation of a system called breakfast serving using ATL to explain how to use the rules the paper proposed to realize the transformation from UML class diagram to HBase database model. The class diagram of the system is shown in Figure 8:

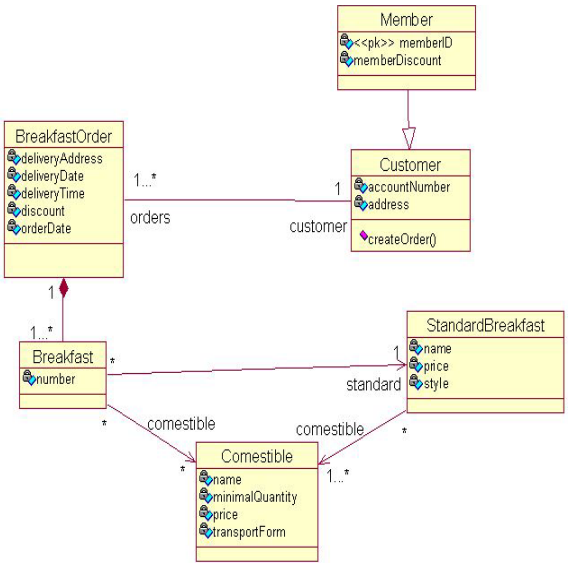

Figure 8. Source model

This system contains six classes. Class BreakfastOrder has “one to many” association relationship with class Customer and has composition relationship with class Breakfast; class Customer has generalization relationship with class Member; class Breakfast and class StandardBreakfast have association relationship with class Comestible. There will create five tables after transformation according to the rules. The target model is generated by using the transformation of ATL. The code of target model is as follow:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI                                    xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI" xmlns="HBase">
    <Table name="StandardBreakfast">
        <rowkey name="(standardID)"/>
        <timestamp/>
        <colfamily name="info">
            <column name="name"/>
            <column name="price"/>
            <column name="style"/>
        </colfamily>
        <colfamily name="Comestibleforeign">
            <column name="comestibleID"/>
        </colfamily>
        <colfamily name="Breakfastforeign">
            <column name="breakfastID"/>
        </colfamily>
     </Table>
     ……
    </xmi: XMI>
```

The data model of HBase can be seen in the Ecore Model Editor of EMF, which is shown in Figure 9. There are five tables and every table contains row key, timestamp, column families and columns in the data model of HBase. It’s an integrated database system. There are six classes in the breakfast serving system, but class Member is a subclass of class Customer. All the elements of subclass are transformed into the elements of parent class, so the target model has five tables. And every attribute of every class has been transformed into related element of HBase database according to the rules at last.

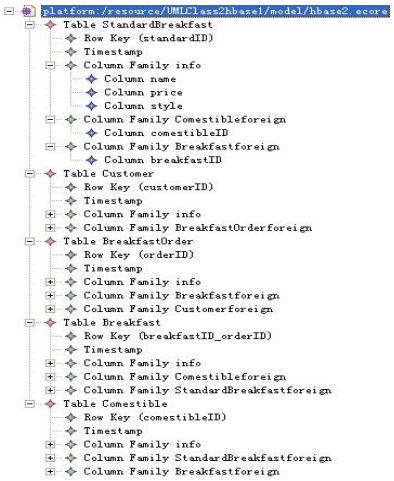

Figure 9. Target model

V. CONCLUTION

The paper proposes a method of how to transform UML class diagram into HBase database model based on the idea of MDA. The main work of the paper is proposing the mapping rules of every element of the meta-model of UML class diagram and HBase database model to realize the transformation from class diagram to HBase data model after building the meta-model of UML class diagram and HBase database model. However, the faults which will affect the results of transformation don’t be considered in the paper and the mapping rules of dependency relationship can’t be analyzed in the paper because it is to complex. In the next step, we will consider the faults probably generated during the transformation, and try to add the function of debugging into the transforming program. We also will consider more and more complex models to complete the mapping between the meta-model of class diagram and HBase database model, and use the mapping rules to realize a special system based on HBase database in practice.

REFERENCES

[1] Anneke Kleppe. MDA Explained: The Practice and Promise of The Model Driven Architecture [M]. Boston: Pearson Education, Inc. 2003: 1-31.

[2] Chongxin Li. Transforming relational database into HBase: A case study [C]. Beijing: Software Engineering and Service Sciences (ICSESS), 2010 IEEE International Conference on Beijing, 2010: 683-687.

[3] Xinfu Xiong. Research and Implementation of Relational Database Model based on UML [D]. Chengdu: University of Electronic Science and Technology. 2011.

[4] Lars George. Guide of HBase [M]. Nanjing : Publication of Southeast University, 2012.

[5] Tom White. Guide of Hadoop[M]. Beijing: Publication of Tsinghua University, 2011.

[6] David S. Frankel. Model Driven Architecture: Applying MDA to Enterprise Computing [M]. New Jesey: John Wiley&Sons, Inc. 2003: 95-109.

[7] Zhang Bo, Li Yafen. Research of relational model transformation based on ATL [C]. Beijing: Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on Beijing, 2012, 115-118.

[8] France R B. Model-driven development using UML2.0: promises and pitfalls [J].Computer, 2006, 39(2):59-66.

[9] ATLAS group LINA&INRIA Nantes. Atlas Transformation Language ATL User Manual. Version 0.7.2006

[10] Eclipse. ATL/Concepts. http://wiki.eclipse.org/ATL/Concepts.